WL-TR-96-2018

REDUCTION OF DATA FROM HEAT-FLUX GAUGES--A DOCUMENTATION OF THE MIT ACQ CODE AND AN ADAPTATION TO SINGLE-SIDED GUAGES

M. M. Weaver
J. Moselle
M. Dunn
G. Guenette

Calspan Corp
Advanced Technology Center
PO Box 400
Buffalo NY 14225

MARCH 1994

FINAL

DTIC QUALITY INSPECTED 4

Approved for public release; distribution unlimited

AERO PROPULSION & POWER DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7251

19961011 119

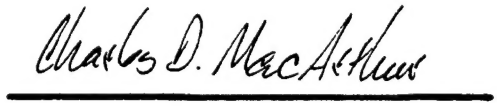
NOTICE

WHEN GOVERNMENT DRAWINGS, SPECIFICATIONS, OR OTHER DATA ARE USED FOR ANY PURPOSE OTHER THAN IN CONNECTION WITH A DEFINITELY GOVERNMENT-RELATED PROCUREMENT, THE UNITED STATES GOVERNMENT INCURS NO RESPONSIBILITY OR ANY OBLIGATION WHATSOEVER. THE FACT THAT THE GOVERNMENT MAY HAVE FORMULATED OR IN ANY WAY SUPPLIED THE SAID DRAWINGS, SPECIFICATIONS, OR OTHER DATA, IS NOT TO BE REGARDED BY IMPLICATION, OR OTHERWISE IN ANY MANNER CONSTRUED, AS LICENSING THE HOLDER, OR ANY OTHER PERSON OR CORPORATION; OR AS CONVEYING ANY RIGHTS OR PERMISSION TO MANUFACTURE, USE, OR SELL ANY PATENTED INVENTION THAT MAY IN ANY WAY BE RELATED THERETO.

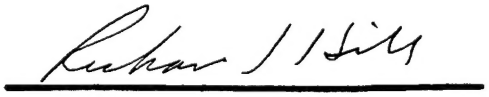
THIS REPORT IS RELEASABLE TO THE NATIONAL TECHNICAL INFORMATION SERVICE (NTIS). AT NTIS, IT WILL BE AVAILABLE TO THE GENERAL PUBLIC, INCLUDING FOREIGN NATIONS.

THE TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.

CHRISTIAN E. RANDELL, Lt, USAF
Turbine Design Engineer
Turbine Engine Division



CHARLES D. MacARTHUR
Chief, Turbine Branch
Turbine Engine Divsion



RICHARD J. HILL
Chief of Technology
Turbine Engine Division
Aero Propulsion & Power Directorate

IF YOUR ADDRESS HAS CHANGED, IF YOU WISH TO BE REMOVED FROM OUR MAILING LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION PLEASE NOTIFY WL/POTT WPAFB OH 45433-7251 HELP MAINTAIN A CURRENT MAILING LIST.

COPIES OF THIS REPORT SHOULD NOT BE RETURNED UNLESS RETURN IS REQUIRED BY SECURITY CONSIDERATIONS, CONTRACTUAL OBLIGATIONS, OR NOTICE ON A SPECIFIC DOCUMENT.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Mar 94	Final

4. TITLE AND SUBTITLE
Reduction of Data from Heat-Flux Gauges -- a Documentation of the MIT ACQ Code and an Adaptation to Single-Sided Gauges

5. FUNDING NUMBERS
C: F33615-88-C-2825
PE: 62203F
PR: 3066
TA: 06
WU: 84

6. AUTHOR(S)
M. M. Weaver M. Dunn
J. Moselle G. Guenette

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Calspan Corp
Advanced Technology Ctr
PO Box 400
Buffalo NY 14225

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Aero Propulsion & Power Directorate
Wright Laboratory
Air Force Materiel Command
Wright-Patterson AFB OH 45433-7251
POC: Lt Christian Randell, WL/POTT, WPAFB OH, 513-255-3150

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
WL-TR-96-2018

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Massachusetts Institute of Technology (MIT) ACQ code for the reduction of data from heat-flux gauges has been adapted for use in the Turbine Research Facility (TRF) Data Acquisition System (DAS). MIT developed the code to determine the fluctuating (AC) heat-flux component from their double-sided gauges. Here the code is adapted to single-sided gauges with the capability to determine both mean (DC) and fluctuating (AC) heat-flux levels. The code is verified using several test cases and thoroughly documented.

14. SUBJECT TERMS
Heat Flux Gauge Data, Double and Single -Sided, Reduction, Heat Flux Gauges

15. NUMBER OF PAGES
88

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

TABLE OF CONTENTS

PAGE

LIST OF FIGURES

FIGURE PAGE

Figure Page

LIST OF SYMBOLS

C -- capacitance

C' -- normalized capacitance

C_t -- total analogue capacitance

c -- capacitance per unit length

C_p -- specific heat of substrate

d -- substrate thickness

FF -- forcing function

f_s -- sample frequency

I -- current

k -- thermal conductivity of substrate

n -- number of nodes in analogue

[Q] -- matrix of eigenvectors

q_{ij} -- element of eigenvector matrix

$\dot{q}$ -- heat-flux

R -- resistance

R' -- normalized resistance

R_t -- total analogue resistance

r -- resistance per unit length

T -- temperature

t -- time

U -- transformed voltage

V -- voltage

x -- depth into substrate from upper surface

y -- coordinate of solution domain

γ -- logarithmic geometric factor

λ_i -- eigenvalue

ρ -- density of substrate

ω -- frequency in rad/sec

SECTION 1 -- INTRODUCTION, HEAT-FLUX DATA REDUCTION

This report documents the data reduction code referred to as ACQ. The code was originally developed at the Massachusetts Institute of Technology (MIT) in 1984 for the United States Air Force to reduce data from MIT's (then recently developed) double-sided heat-flux gauges. The ACQ code has been used at MIT for nearly ten years.

The study conducted at Calspan to document ACQ has several different facets. The first facet examines the theory and methodology applied in the data reduction code. The second facet of the study ascertains the code's capabilities and accuracy. These first two facets of the study yielded an understanding of the code which made it possible to apply the code in a more user friendly and versatile way. The last facet of the study documents this application of the code within the Data Acquisition System (DAS).

The ACQ data reduction code is capable of reducing data from both single-sided (semi-infinite) and double-sided (finite) heat-flux gauges. Single-sided gauges, sometimes referred to as thin-film semi-infinite gauges, were in use prior to the development of the double-sided gauge at MIT and are still being used at many laboratories with success.

The semi-infinite gauges consist of a thin-film resistance thermometer bonded to an insulating substrate. It is assumed that the substrate appears infinitely thick to thermal waves propagating from the surface. Therefore, the time-dependent heat-flux can be inferred using a one dimensional heat conduction model. The semi-infinite gauges are typically constructed of platinum painted onto a sufficiently thick piece of Pyrex glass which is then inserted into the metal airfoil. Although, in some instances the entire airfoil is constructed from the insulating material; usually a machinable ceramic.

The double-sided gauges are a relative of the semi-infinite gauges for which a second thin-film resistance thermometer is bonded on the opposite side of the insulating substrate. This eliminates the need to assume that the substrate is semi-infinite because the measured bottom temperature is the second boundary condition. Therefore, the substrate can be made very thin making it possible to cover the airfoil surface with a layer of the substrate material. A schematic cross section of a double-sided heat-flux gauge is given in Figure 1.1.

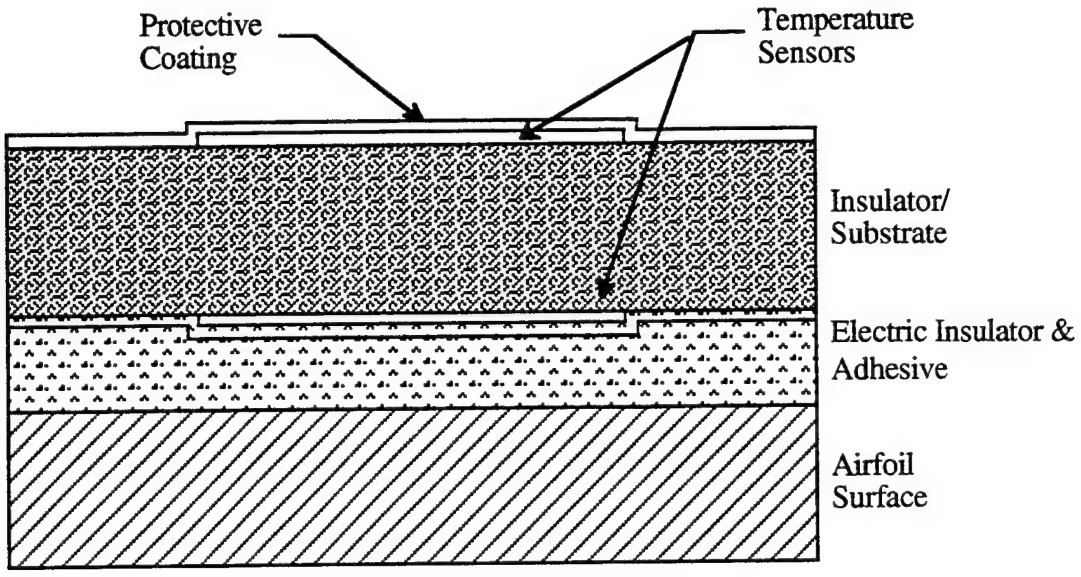


Figure 1.1 -- Schematic Cross Section of Double-Sided Gauge

The double-sided gauges act as a thermal shunt below a certain frequency where the temperature difference across the insulator is a direct measure of the heat-flux. The shunt cut-off frequency increases as the insulator thickness is reduced. Conversely, above another frequency the substrate appears semi-infinite to the upper sensor and a one dimensional assumption can be used to infer the heat-flux. These limiting cases are not in themselves applicable in the reduction of data because most of the signal frequencies lie

between the limiting frequencies. Therefore, a numerical technique is used to reconstruct the entire frequency domain from DC to 100 kHz.

In the numerical technique, the time unsteady one dimensional heat conduction through the gauge substrate is modeled as current flow through a discrete resistance-capacitance network. A schematic of this analogue is given in Figure 1.2. The current through the first resistor represents the surface heat-flux for the measured surface temperatures which are the input voltages to the circuit. The resistor and capacitor elements are analogous to the thermal properties of the substrate. The circuit elements are logarithmically distributed through the network to match the insulator attenuation characteristics at high frequencies to increase the numerical efficiency.

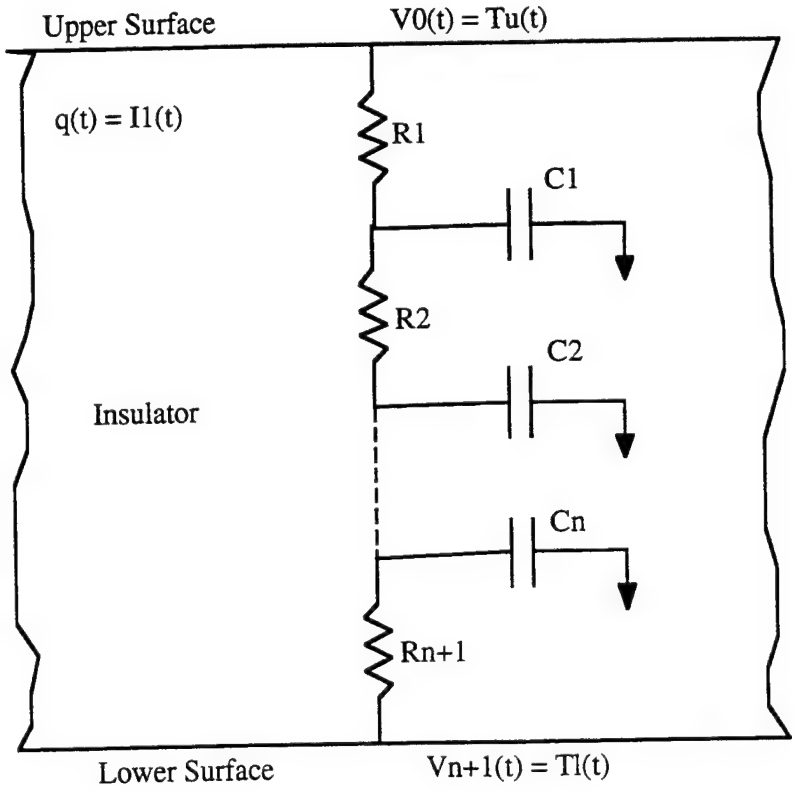


Figure 1.2 -- Electric Analogue Schematic of Double-Sided Gauge

The theory applied in the data reduction code is detailed in Section 2. The electric analogue is developed with comparison to an alternative finite-difference approach. The R-C circuit construction is outlined along with the solution procedure for the resulting system of equations.

Section 3 documents the verification of the ACQ code. MIT's original verification study is duplicated along with additional studies to ascertain the accuracy of the code. The two test cases used by MIT demonstrated the ability of ACQ to produce heat-flux data from double-sided gauge data. A third test case is given to demonstrate how ACQ can be used to reduce single-sided semi-infinite gauge data. The second part of the verification process examines how anomalies in the recorded data affect the code's solution. Truncation error due to low A/D resolution and random signal noise are both investigated.

Section 4 outlines the application of the data reduction code. ACQ has been modified to enable input and output of Data Acquisition System (DAS) format files. The program can be operated in two modes: interactive and automatic. The interactive mode provides the user with a convenient tool for generating heat-flux time histories for one or two gauges. The automatic mode is more like a batch process which not only performs multiple gauge heat-flux calculations, but also places pertinent information into the run log files. Proper application of the code is addressed which requires some consideration in the design of the experiment.

SECTION 2 -- DATA REDUCTION THEORY

The heat-flux time history is deduced from the recorded gauge surface temperatures. The gauge surface temperatures are used as boundary conditions in solving the diffusion equation through the substrate between the temperature sensors. The thermal diffusion equation is given by Equation 2.1 and the corresponding heat-flux is given by Equation 2.2.

$$\frac{\partial T}{\partial t} = \frac{k}{\rho C_p}\frac{\partial^2 T}{\partial x^2} \tag{2.1}$$

$$\dot{q} = -k\frac{\partial T}{\partial x} \tag{2.2}$$

2.1 -- Electric Analogue

An electric analogue to the thermal diffusion equation is used by Norton [1985] in the data reduction code ACQ supplied to Calspan by MIT. This analogue is convenient because it follows the theory developed by Burd and Doe [1980] under the direction of Oldfield. The predecessor of the electric analogue circuit described by Burd and Doe was developed by Skinner [1960]. In this theory the gauge substrate is analogous to a resistance-capacitance circuit with the diffusion equation taking the form

$$\frac{\partial V}{\partial t} = \frac{1}{rc}\frac{\partial^2 V}{\partial x^2} \tag{2.3}$$

where, r=resistance/unit length, c=capacitance/unit length, $V \equiv T$, $I \equiv \dot{q}$, $c \equiv \rho C_p$, $r \equiv \frac{1}{k}$.

2.1.1 -- Discrete Analogue

A discrete R-C circuit, shown in Figure 2.1, is constructed to discretize the spatial domain. For the semi-infinite case, R_{n+1} is set to a very large value.

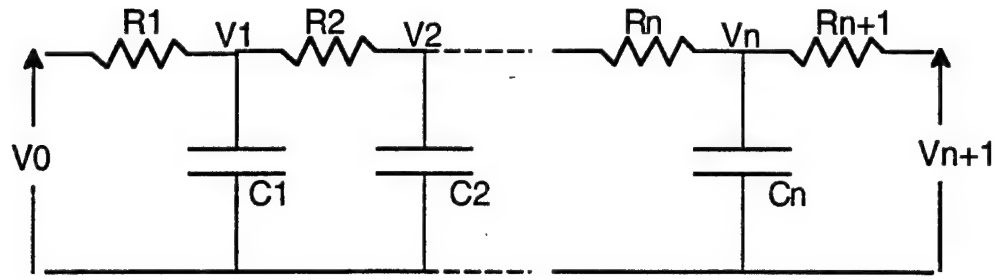


Figure 2.1 -- R-C Circuit Discrete Analogue

Consider the element of this circuit shown in Figure 2.2.

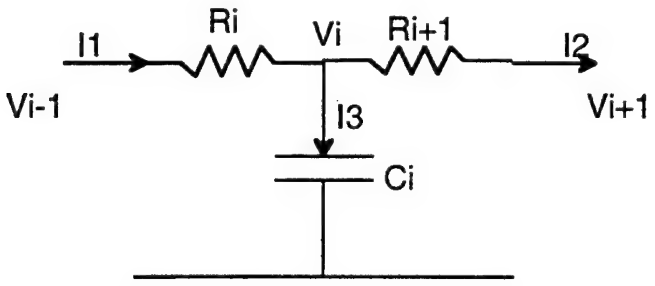


Figure 2.2 -- R-C Circuit Element

From elementary circuit analysis,

$$V_{i-1} - V_i = I_1 R_i$$

$$V_i - V_{i+1} = I_2 R_{i+1}$$

$$I_3 = C_i \frac{dV_i}{dt}$$

$$I_3 = I_1 - I_2$$

$$C_i \frac{dV_i}{dt} = \frac{V_{i-1} - V_i}{R_i} - \frac{V_i - V_{i+1}}{R_{i+1}}$$

rearranging,
$$\frac{dV_i}{dt} = \frac{V_{i-1}}{R_i C_i} - \frac{V_i}{C_i}\left(\frac{1}{R_i} + \frac{1}{R_{i+1}}\right) + \frac{V_{i+1}}{R_{i+1} C_i} \tag{2.4}$$

Equation 2.4 is a first order Ordinary Differential Equation (ODE) for the voltage at each node along the circuit. To demonstrate that Equation 2.4 is truly analogous, an identical ODE is derived from a finite difference analysis of the thermal diffusion equation. Consider the temperature to be known at discrete spatial locations which are not equally spaced. The second order spatial derivative of the temperature is approximated by

$$\frac{\partial^2 T_i}{\partial x^2} = \frac{T_{i-1}}{\Delta x_i \Delta x_i} - \frac{T_i}{\Delta x_i}\left(\frac{1}{\Delta x_i} + \frac{1}{\Delta x_{i+1}}\right) + \frac{T_{i+1}}{\Delta x_{i+1} \Delta x_i}$$

where,
$$\Delta x_i = x_i - x_{i-1}$$

$$\Delta x_{i+1} = x_{i+1} - x_i$$

substituting into Equation 2.1,

$$\frac{\partial T_i}{\partial t} = \frac{T_{i-1}}{(\rho C_p \Delta x_i)\frac{\Delta x_i}{k}} - \frac{T_i}{\rho C_p \Delta x_i}\left(\frac{1}{\frac{\Delta x_i}{k}} + \frac{1}{\frac{\Delta x_{i+1}}{k}}\right) + \frac{T_{i+1}}{\frac{\Delta x_{i+1}}{k}(\rho C_p \Delta x_i)} \tag{2.5}$$

Equation 2.5 is analogous to Equation 2.4 with the capacitance equivalent to the total thermal capacity of a depth Δx_i, and the resistance equivalent to the total thermal impedance of a depth Δx_i.

$$C_i = \rho C_p \Delta x_i$$

$$R_i = \frac{\Delta x_i}{k}$$

The discrete R-C circuit analysis results in exactly the same equation as the finite difference analysis of the thermal diffusion equation. Thus, we are solving the correct analogous equation, Equation 2.4.

2.1.2 -- Matrix Theory for System of Equations

The application of Equation 2.4 to all the nodes of the circuit produces a set of coupled first order ODE's. The voltages at the ends of the circuit are input and analogous to the substrate surface temperatures. Applying Equation 2.4 at the endpoints,

$$\frac{dV_1}{dt} = \frac{V_0}{R_1C_1} - \frac{V_1}{C_1}\left(\frac{1}{R_1} + \frac{1}{R_2}\right) + \frac{V_2}{R_2C_1}$$

$$\frac{dV_n}{dt} = \frac{V_{n-1}}{R_nC_n} - \frac{V_n}{C_n}\left(\frac{1}{R_n} + \frac{1}{R_{n+1}}\right) + \frac{V_{n+1}}{R_{n+1}C_n}$$

Casting the set of equations into vector form,

$$\frac{d\vec{V}}{dt} = [A]\vec{V} + \frac{V_0}{R_1C_1}\vec{k}_1 + \frac{V_{n+1}}{R_{n+1}C_n}\vec{k}_n \qquad (2.6)$$

where,
$$\vec{V} = \begin{Bmatrix} V_1 \\ V_2 \\ \bullet \\ V_n \end{Bmatrix}, \quad \vec{k}_1 = \begin{Bmatrix} 1 \\ 0 \\ \bullet \\ 0 \end{Bmatrix}, \quad \vec{k}_n = \begin{Bmatrix} 0 \\ 0 \\ \bullet \\ 1 \end{Bmatrix}$$

V_0 is analogous to the upper surface temperature and V_{n+1} is analogous to the lower surface temperature.

$$[A]=\begin{bmatrix} -\frac{1}{C_1}\left(\frac{1}{R_1}+\frac{1}{R_2}\right) & \frac{1}{C_1R_2} & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ \frac{1}{C_2R_2} & -\frac{1}{C_2}\left(\frac{1}{R_2}+\frac{1}{R_3}\right) & \frac{1}{C_2R_3} & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & \cdot & \cdot & 0 & \cdot & \cdot & 0 \\ \cdot & 0 & \cdot & \cdot & \cdot & 0 & \cdot & 0 \\ \cdot & \cdot & 0 & \cdot & \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & 0 & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & 0 & 0 & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{C_nR_n} & -\frac{1}{C_n}\left(\frac{1}{R_n}+\frac{1}{R_{n+1}}\right) \end{bmatrix}$$

This coefficient matrix is cast into a symmetric matrix using the transformation

$$V_i=\frac{U_i}{\sqrt{C_i}} \tag{2.7}$$

Equation 2.4 becomes

$$\frac{dU_i}{dt}=\frac{U_{i-1}}{R_i\sqrt{C_iC_{i-1}}}-\frac{U_i}{C_i}\left(\frac{1}{R_i}+\frac{1}{R_{i+1}}\right)+\frac{U_{i+1}}{R_{i+1}\sqrt{C_iC_{i+1}}}$$

The set of equations in matrix form become

$$\frac{d\vec{U}}{dt}=[B]\vec{U}+\frac{V_0}{R_1\sqrt{C_1}}\vec{k}_1+\frac{V_{n+1}}{R_{n+1}\sqrt{C_n}}\vec{k}_n \tag{2.8}$$

where [B] is a tri-diagonal matrix with the following elements

$$b_{ii}=-\frac{1}{C_i}\left(\frac{1}{R_i}+\frac{1}{R_{i+1}}\right)$$

$$b_{i,i+1}=b_{i+1,i}=\frac{1}{R_{i+1}\sqrt{C_iC_{i+1}}}$$

Note that V_0 and V_{n+1} are not transformed to U values because C_0 and C_{n+1} are undefined.

The set of ODE's are uncoupled by defining a new set of coordinates, y, such that

$$\vec{U} = [Q]\vec{y} \tag{2.9}$$

where [Q] is the matrix of eigenvectors for [B]. Substituting into Equation 2.8,

$$[Q]\frac{d\vec{y}}{dt} = [B][Q]\vec{y} + \frac{V_0}{R_1\sqrt{C_1}}\vec{k}_1 + \frac{V_{n+1}}{R_{n+1}\sqrt{C_n}}\vec{k}_n$$

Multiply through by $[Q]^{-1}$

$$[Q]^{-1}[Q]\frac{d\vec{y}}{dt} = [Q]^{-1}[B][Q]\vec{y} + [Q]^{-1}\frac{V_0}{R_1\sqrt{C_1}}\vec{k}_1 + [Q]^{-1}\frac{V_{n+1}}{R_{n+1}\sqrt{C_n}}\vec{k}_n \tag{2.10}$$

Now
$$[Q]^{-1}[B][Q] = \begin{bmatrix} \lambda_1 & 0 & \cdot & \cdot & 0 \\ 0 & \lambda_2 & 0 & \cdot & 0 \\ 0 & 0 & \cdot & 0 & 0 \\ 0 & \cdot & 0 & \cdot & 0 \\ 0 & \cdot & \cdot & 0 & \lambda_n \end{bmatrix}$$

where the diagonal elements are the eigenvalues of [B]. Since [B] is symmetric, its eigenvectors are orthogonal, hence

$$[Q]^{-1} = [Q]^T = \begin{bmatrix} q_{11} & q_{21} & q_{31} & \cdot & \cdot & \cdot & q_{n1} \\ q_{12} & \cdot & & & & & q_{n2} \\ q_{13} & & \cdot & & & & q_{n3} \\ \cdot & & & \cdot & & & \cdot \\ \cdot & & & & \cdot & & \cdot \\ \cdot & & & & & \cdot & \cdot \\ q_{1n} & q_{2n} & q_{3n} & \cdot & \cdot & \cdot & q_{nn} \end{bmatrix}$$

Then, if y_i is the i^{th} element of column vector y, Equation 2.10 gives,

$$\frac{dy_i}{dt} = \lambda_i y_i + q_{1,i}\frac{V_0}{R_1\sqrt{C_1}} + q_{n,i}\frac{V_{n+1}}{R_{n+1}\sqrt{C_n}} \tag{2.11}$$

Thus, Equation 2.11 is the uncoupled ODE to be solved at each of the circuit nodes.

2.2 Solution Procedure

The theory outlined up to this point has required specific resistance and capacitance values. Since the thermal properties and thicknesses will vary from gauge to gauge, so will the specific resistance and capacitance values. Thus to avoid evaluating the eignevalues and eigenvectors for each gauge, a normalized R-C circuit is set up with

$$R_i' = \frac{R_i}{R_1} \quad \& \quad C_i' = \frac{C_i}{C_1} \tag{2.12}$$

Then Equation 2.11 becomes

$$\frac{dy_i}{dt} = \frac{1}{R_1 C_1}\left\{\lambda_i y_i + q_{1,i}\frac{V_0}{R_1'\sqrt{C_1'}} + q_{n,i}\frac{V_{n+1}}{R_{n+1}'\sqrt{C_n'}}\right\} \tag{2.13}$$

and the transformation given by Equation 2.7 becomes

$$V_i = \frac{U_i}{\sqrt{C_i'}} \tag{2.14}$$

To have the same eigenvalues and eigenvectors, each gauge analogue in the series must have the same number of circuit nodes (stages) and the same geometric variation in the circuit components. The specific thermal and physical properties of each gauge are introduced through R_1 and C_1.

2.2.1 -- R-C Component Selection

Consider the circuit shown in Figure 2.1 to consist of a series of 'T' sections as shown in Figure 2.3.

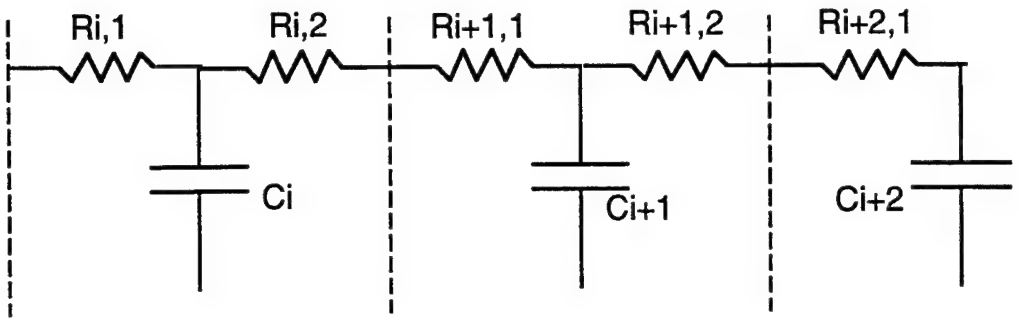


Figure 2.3 -- R-C Circuit as a Series of 'T' Sections

Let β be the resistor matching coefficient

$$\beta = \frac{R_{i,1}}{R_{i,1} + R_{i,2}} \tag{2.15}$$

The ratio R/C for each 'T' section must be equal to r/c, therefore

$$\frac{r}{c} = \frac{R_{i,1} + R_{i,2}}{C_i} \tag{2.16a}$$

$$\frac{r}{c} = \frac{R_{i+1,1} + R_{i+1,2}}{C_{i+1}} \tag{2.16b}$$

The geometric variation of the capacitance and resistance is chosen to be logarithmic to simulate the attenuation characteristics of the gauge substrate.

$$C_{i+1} = \gamma C_i \tag{2.17}$$

$$\frac{R_{i,2}}{R_{i,1}} = \frac{R_{i+1,1}}{R_{i,2}} = \frac{R_{i+1,2}}{R_{i+1,1}} = \alpha \tag{2.18}$$

From Equations 2.16 and 2.17,

$$\frac{R_{i+1,1} + R_{i+1,2}}{R_{i,1} + R_{i,2}} = \gamma \tag{2.19}$$

Then, with Equation 2.18 and 2.19

$$\frac{\alpha^2 + \alpha^3}{1 + \alpha} = \gamma \quad \therefore \quad \gamma = \alpha^2 \tag{2.20}$$

Replacing $R_{i,2}$ in Equation 2.16a, we obtain the relationship for the first resistance in the 'T' section.

$$R_{i,1} = \left(\frac{r}{c}\right)\frac{C_i}{1+\sqrt{\gamma}} \tag{2.21}$$

Combining Equations 2.15, 2.16a and 2.21 gives

$$\beta C_i\left(\frac{r}{c}\right) = \left(\frac{r}{c}\right)\frac{C_i}{1+\sqrt{\gamma}} \quad \therefore \quad \beta = \frac{1}{1+\sqrt{\gamma}} \tag{2.22}$$

The relationship for the second resistance in the 'T' section can be written as

$$R_{i,2} = \left(\frac{\sqrt{\gamma}}{1+\sqrt{\gamma}}\right)\left(\frac{r}{c}\right)C_i \tag{2.23}$$

Returning to the original 'L' section circuit shown in Figure 2.1, the resistances can be defined as follows.

$$R_1 = R_{i,1} = \left(\frac{C_1}{1+\sqrt{\gamma}}\right)\left(\frac{r}{c}\right) \tag{2.24}$$

$$R_{n+1} = R_{n,2} = \left(\frac{C_n\sqrt{\gamma}}{1+\sqrt{\gamma}}\right)\left(\frac{r}{c}\right) \tag{2.25}$$

$$R_{i+1} = R_{i,2} + R_{i+1,1} = \frac{C_{i+1}}{\sqrt{\gamma}}\left(\frac{r}{c}\right) \tag{2.26}$$

2.2.1.1 -- Logarithmic Geometric Factor

The logarithmic geometric factor, γ, is determined through a required bandwidth consideration. The Bode plot for a finite analogue is given in Figure 2.4.

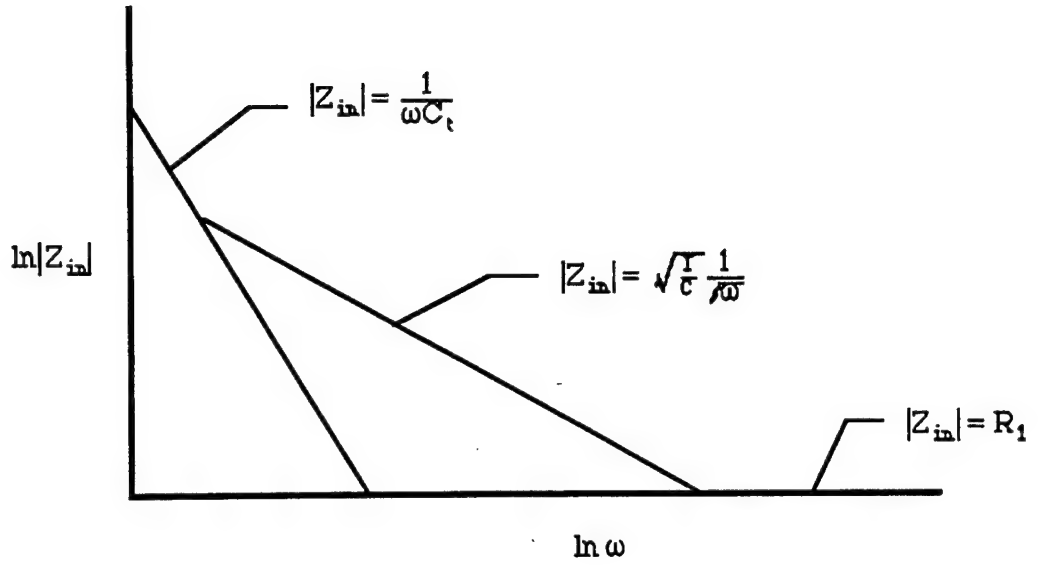


Figure 2.4 -- Finite Analogue Bode Plot

In the normal operating range an analogue's impedance should ideally satisfy

$$|Z_{in}| = \sqrt{\frac{r}{c}} \frac{1}{\sqrt{\omega}}$$

At high frequencies the first capacitor shorts, so

$$|Z_{in}| = R_1$$

At low frequencies the impedance of the capacitors swamps the resistors, so

$$|Z_{in}| = \frac{1}{\omega C_t}$$

where C_t is the total capacitance of the analogue. The useful bandwidth is taken as the intersections of these three curves.

$$R_1 = \sqrt{\frac{r}{c}} \frac{1}{\sqrt{\omega_{max}}} \text{ and } \frac{1}{\omega_{min} C_t} = \sqrt{\frac{r}{c}} \frac{1}{\sqrt{\omega_{min}}}$$

Therefore,

$$\omega_{max} = \frac{r}{c} \frac{1}{R_1^2} \text{ and } \omega_{min} = \frac{c}{r} \frac{1}{C_t^2} \qquad (2.27)$$

Defining the forcing function and evaluating the bandwidth, we get

$$FF = \sqrt{\frac{\omega_{max}}{\omega_{min}}}$$

then

$$FF = \frac{r}{c}\frac{C_t}{R_1}$$

From the resistance-capacitance relationship defined earlier in Equation 2.24

$$FF = \left(1+\sqrt{\gamma}\right)\frac{C_t}{C_1} \tag{2.28}$$

The total capacitance is the sum of the individual capacitances which are given in relation to C_1 by Equation 2.17.

$$C_t = \sum_{i=1}^{n} C_i = \sum C_1 + \gamma C_1 + \gamma^2 C_1 + \cdots + \gamma^{n-1} C_1$$

$$C_t = C_1 \sum 1 + \gamma + \gamma^2 + \cdots + \gamma^{n-1}$$

$$\therefore \frac{C_t}{C_1} = \frac{\gamma^n - 1}{\gamma - 1}$$

Substituting into Equation 2.28 yields,

$$FF = \frac{\left(1+\sqrt{\gamma}\right)\left(\gamma^n - 1\right)}{\gamma - 1} \tag{2.29}$$

Equation 2.29 is solved iteratively for the geometric factor, γ, with the desired bandwidth and specified number of stages, n.

The maximum frequency is taken as one half the sample frequency for both the semi-infinite and the finite cases. The minimum frequency is taken to be the frequency at which the thermal wave propagation depth equals the sensor thickness. This minimum frequency is determined according to the following analysis.

$$e^{-x\sqrt{\frac{\rho C_p \omega}{k}}} = \frac{1}{e} \text{ for } x = d$$

where d is the gauge substrate thickness. Solving for the frequency,

$$\omega_{min} = \left(\frac{k}{d}\right)^2 \frac{1}{\rho C_p k} \tag{2.30}$$

Since the object is to set up a normalized analogue, the $\rho C_p k$ and k/d values used are those for the first sensor to be analyzed.

2.2.2 -- Setting up the Normalized Circuit

The normalized resistance and capacitance values used are

$$R_1' = 1 \text{ and } C_1' = 1$$

$$C_i' = \gamma C_{i-1}' \tag{2.31}$$

$$R_i' = \sqrt{\gamma}\left(1+\sqrt{\gamma}\right)C_{i-1}' \tag{2.32}$$

$$R_{n+1}' = \sqrt{\gamma}C_n' \tag{2.33}$$

The normalized total resistance and capacitance are calculated as

$$R_t' = \sum_{i=1}^{m} R_i' \tag{2.34}$$

$$C_t' = \sum_{i=1}^{n} C_i' \tag{2.35}$$

where m=n for semi-infinite and n+1 for finite. These totals are required to determine R_1 and C_1 for each individual gauge.

The normalized [B] matrix is constructed and stored as two column matrices [D] and [E]. The diagonal elements of [B] are contained in [D],

$$d_i = -\frac{1}{C_i'}\left(\frac{1}{R_i'} + \frac{1}{R_{i+1}'}\right)$$

and the off-diagonal elements are contained in [E].

$$e_{i+1} = \frac{1}{R'_{i+1}\sqrt{C'_i C'_{i+1}}}$$

The eigenvectors and eigenvalues of [B] are calculated by a subroutine which has [D], [E] and an identity matrix [I] passed to it. Upon return, the [D] column matrix contains the eignevalues and the identity matrix contains the eigenvectors.

2.2.3 -- Dimensional Solution

The physical properties of each gauge are introduced to determine the heat-flux corresponding to the surface temperature history of each particular gauge.

2.2.3.1 -- Gauge Physical Properties

The physical properties are introduced through R_1 and C_1 for each gauge. For a finite analogue, the analysis yields the following relationships:

by definition $$R_i \equiv \frac{\Delta x_i}{k}$$

$$C_i \equiv \rho C_p \Delta x_i$$

$$\therefore R_t = \sum R_i = \sum R'_i R_1 = R'_t R_1 = \frac{d}{k}$$

$$C_t = \sum C_i = \sum C'_i C_1 = C'_t C_1 = \rho C_p d$$

$$\therefore R_1 = \frac{d}{k}\frac{1}{R'_t} \tag{2.36}$$

$$C_1 = \frac{d}{k}\frac{\rho C_p k}{C'_t} \tag{2.37}$$

For the semi-infinite analogue R_1 and C_1 are set to give the desired maximum frequency. Recall Equation 2.27,

$$\omega_{max} = \frac{r}{c}\frac{1}{R_1^2}$$

where

$$\frac{r}{c} \equiv \frac{1}{\rho C_p k}$$

$$\therefore R_1 = \frac{1}{\sqrt{\rho C_p k \omega_{max}}} \tag{2.38}$$

Recall Equation 2.24,

$$R_1 = \frac{r}{c}\frac{C_1}{\left(1+\sqrt{\gamma}\right)}$$

$$\therefore C_1 = \frac{\left(1+\sqrt{\gamma}\right)\sqrt{\rho C_p k}}{\sqrt{\omega_{max}}} \tag{2.39}$$

2.2.3.2 -- Integration of ODE

The solution is started by setting the initial condition at each node in the circuit. The original version of the code (1.10--10/30/84) did not implement a physically correct initial condition, but rather set the initial temperature of the substrate equal to zero. This simplified approach was accomplished by setting all the y_i's equal to zero to start the integration. The simplified initial condition caused a large initial spike in the calculated heat-flux value which quickly decayed into the steady-state solution. This physically incorrect initial transient was corrected by implementing a more realistic initial condition.

Non-uniform initial conditions are considered because it is often desirable to only reduce a portion of the recorded data. If the data reduction begins at a time after the tunnel flow begins, the substrate is no longer at a uniform temperature. A non-uniform initial condition can be estimated for a finite thickness case because the lower surface temperature is known. However, there is no accurate method to specify a non-uniform initial condition for the semi-infinite thickness case. Therefore, the following two approaches are used.

2.2.3.2.1 -- Finite Thickness Circuit Initial Conditions

To account for non-uniform initial conditions, a linear distribution between the known upper and lower surface temperatures is assumed. The linear distribution corresponds physically to steady state heat conduction through the gauge substrate.

$$T(x) = T_0 + (T_{n+1} - T_0)\frac{x}{d}$$

where T_0 is the upper surface temperature at the start time and T_{n+1} is the bottom surface temperature at the start time. In terms of the analogue,

$$R_i = \frac{\Delta x_i}{k}$$

$$\therefore \frac{\Delta x_i}{d} = \frac{k}{d} R_i$$

and,

$$\frac{x_m}{d} = \frac{k}{d}\sum_{i=1}^{m} R_i = R_1 \frac{k}{d}\sum_{i=1}^{m} R_i'$$

Therefore, the initial voltage at any circuit node "m" is

$$V_m = V_0 + (V_{n+1} - V_0) R_1 \frac{k}{d}\sum_{i=1}^{m} R_i' \qquad (2.40)$$

where m = 1, 2, 3, . . ., n.

These initial voltages are then transformed into y coordinates to start the integration. Recall the first transformation to a symmetric coefficient matrix given by Equation 2.14.

$$U_i = V_i\sqrt{C_i'}$$

The second transformation to the y coordinate is given by Equation 2.9.

$$\bar{y} = [Q]^{-1}\bar{U}$$

where $[Q]^{-1}$ is the transformed matrix of eigenvectors. The transformation from a voltage at node "m" to the corresponding y_m is given by

$$y_m = \sum_{i=1}^{n} q_{i,m} \sqrt{C_i} V_i \tag{2.41}$$

where again m = 1, 2, 3, . . ., n.

2.2.3.2.2 -- Semi-Infinite Thickness Circuit Initial Conditions

For the semi-infinite case the integration must be started at a time when the initial temperature distribution in the substrate is accurately known. For all practical cases this is at a time before the experiment is initiated when the substrate is at a known uniform temperature. In this case the initial y's are given as

$$y_m = V_0\big|_{t=0} \sum_{i=1}^{n} q_{i,m} \sqrt{C_i} \tag{2.42}$$

2.2.3.2.3 -- Integration of ODE

All the ODE's are integrated over each time interval in the upper surface temperature file. For the finite case, a lower surface temperature is read at the same time. The integration is carried out with a fourth order Runge-Kutta method where

$$\frac{dy}{dt} = f(t,y)$$

The four steps are:

$$An = f(t,y)dt$$

$$Bn = f\left(t + \frac{dt}{2}, y + \frac{An}{2}\right)$$

$$Cn = f\left(t + \frac{dt}{2}, y + \frac{Bn}{2}\right)$$

$$Dn = f(t + dt, y + Cn)$$

and finally,

$$y(t+dt) = y(t) + \frac{An + 2Bn + 2Cn + Dn}{6}$$

This is coded as

$$An_i = \frac{dt}{R_1 C_1}\left\{\lambda_i y_i + \frac{q_{1,i} V_0(t)}{R_1'\sqrt{C_1'}} + \frac{q_{n,i} V_{n+1}(t)}{R_{n+1}'\sqrt{C_n'}}\right\}$$

$$Bn_i = \frac{dt}{R_1 C_1}\left\{\lambda_i\left(y_i + \frac{An}{2}\right) + \frac{q_{1,i}\left[\frac{V_0(t)+V_0(t+dt)}{2}\right]}{R_1'\sqrt{C_1'}} + \frac{q_{n,i}\left[\frac{V_{n+1}(t)+V_{n+1}(t+dt)}{2}\right]}{R_{n+1}'\sqrt{C_n'}}\right\}$$

$$Cn_i = \frac{dt}{R_1 C_1}\left\{\lambda_i\left(y_i + \frac{Bn}{2}\right) + \frac{q_{1,i}\left[\frac{V_0(t)+V_0(t+dt)}{2}\right]}{R_1'\sqrt{C_1'}} + \frac{q_{n,i}\left[\frac{V_{n+1}(t)+V_{n+1}(t+dt)}{2}\right]}{R_{n+1}'\sqrt{C_n'}}\right\}$$

$$Dn_i = \frac{dt}{R_1 C_1}\left\{\lambda_i(y_i + Cn) + \frac{q_{1,i} V_0(t+dt)}{R_1'\sqrt{C_1'}} + \frac{q_{n,i} V_{n+1}(t+dt)}{R_{n+1}'\sqrt{C_n'}}\right\}$$

Thus giving, $y_i(t+dt)$

The final task is to find the voltage at node one to evaluate the heat flux. Working back through the transformations,

$$U_1 = \sum_{i=1}^{n} q_{1,i} y_i$$

$$V_1 = \frac{U_1}{\sqrt{C_1'}}$$

Then, the surface heat-flux value is given by

$$\dot{q} = I = \frac{V_0 - V_1}{R_1} \qquad (2.43)$$

SECTION 3 -- CODE VERIFICATION

Although the data reduction code, known as ACQ, has been in use for nearly ten years at MIT, few details are available on its verification. Therefore, a code verification was conducted at Calspan to document the code's capabilities and accuracy.

Epstein, Guenette, Norton and Yuzhang [1985] reported that the accuracy of ACQ was established by back calculating the surface heat-flux for exact analytical upper and lower surface temperature solutions in two cases: a step change in heat-flux, and a sinusoidal heat-flux variation. They used this same approach to evaluate the number of nodes required for a specific bandwidth and accuracy. It was found that nine nodes yielded agreement to better than 1% over the entire DC to 100 kHz frequency domain including less than 0.3% error in the 5-2000 Hz band.

The code verification carried out here is two fold. The first part revisits the two test cases studied by Epstein et al. [1985] and adds a third semi-infinite case. All the cases use the recommended nine nodes. The second part of the verification process illustrates how anomalies in the recorded signals affect the accuracy of the code's solution. The two anomalies studied are: digitization error due to low analog to digital resolution, and random noise super-imposed onto the signal. The verification process concludes with a comparison of ACQ with Cook/Felderman [1966] using actual data from a semi-infinite Pyrex gauge located on the SSME turbine stage.

3.1 -- Exact Analytical Test Cases

Epstein et al. [1985] provide analytical upper and lower surface temperature solutions for the case of a step change in surface heat-flux, and for the case of a steady harmonic variation in surface heat-flux. These exact temperature profiles are used as input to the code for back calculating the surface heat-flux. A third test case taken from Incropera and DeWitt [1985] for convective heat transfer at the surface of a semi-infinite slab is performed for verification of the semi-infinite calculations.

3.1.1 -- Steady Harmonic Surface Heat-flux Variation

The analytical temperature distribution through a gauge substrate subjected to a steady harmonic surface heat-flux is given below. The surface heat-flux is taken to be

$$\dot{q} = -k\frac{\partial T}{\partial x}\bigg|_{x=0} = \overline{Q} + Q_0\cos(\omega t) \quad \text{for} \quad -\infty < t < +\infty \tag{3.1}$$

where $\overline{Q}$ is the mean or DC heat-flux level; Q_0 is the perturbation in heat-flux about the mean, and ω is the frequency of the perturbation. The mean heat-flux is given by

$$\overline{Q} = \frac{k}{d}\left(\overline{T}(0) - \overline{T}(d)\right) \tag{3.2}$$

where $\overline{T}(0)$ and $\overline{T}(d)$ are the mean upper and lower surface temperatures respectively.

The temperature distribution in the substrate is

$$T(x,t) = \overline{T}(x) + Q_0 M\frac{d}{k}\cos(\omega t - \phi) \quad \text{for} \quad 0 \le x \le d \tag{3.3}$$

where,

$$M = \sqrt{2}\left(\frac{\omega}{\omega_c}\right)^{-\frac{1}{2}}\exp\left[-\frac{1}{2}\left(\frac{\omega}{\omega_c}\right)^{\frac{1}{2}}\frac{x}{d}\right]\left(\frac{A^2+B^2}{C^2+D^2}\right)^{\frac{1}{2}},$$

$$\phi = \frac{1}{2}\left(\frac{\omega}{\omega_c}\right)^{\frac{1}{2}}\frac{x}{d} + \frac{\pi}{4} - \text{Arc}\tan\left(\frac{BC-AD}{AC+BD}\right)$$

A sensor characteristic frequency ω_c is introduced based upon the substrate thickness d, which is defined as

$$\omega_c = \frac{1}{2}\left(\frac{k}{d}\right)^2 \frac{1}{\rho C_p k}$$

The other terms in Equation 3.3 are defined as follows:

$$A = 1 - \Gamma \exp\left[-\left(\frac{\omega}{\omega_c}\right)^{\frac{1}{2}}\left(1-\frac{x}{d}\right)\right]\cos\left[\left(\frac{\omega}{\omega_c}\right)^{\frac{1}{2}}\left(1-\frac{x}{d}\right)\right]$$

$$B = \Gamma \exp\left[-\left(\frac{\omega}{\omega_c}\right)^{\frac{1}{2}}\left(1-\frac{x}{d}\right)\right]\sin\left[\left(\frac{\omega}{\omega_c}\right)^{\frac{1}{2}}\left(1-\frac{x}{d}\right)\right]$$

$$C = 1 + \Gamma \exp\left[-\left(\frac{\omega}{\omega_c}\right)^{\frac{1}{2}}\right]\cos\left(\frac{\omega}{\omega_c}\right)^{\frac{1}{2}}$$

$$D = -\Gamma \exp\left[-\left(\frac{\omega}{\omega_c}\right)^{\frac{1}{2}}\right]\sin\left(\frac{\omega}{\omega_c}\right)^{\frac{1}{2}}$$

where,

$$\Gamma = \frac{\gamma - 1}{\gamma + 1}$$

and

$$\gamma = \frac{\sqrt{(\rho C_p k)_{blade}}}{\sqrt{(\rho C_p k)_{gauge}}}$$

Three test cases are used to study ACQ's accuracy in calculating the steady harmonic surface heat-flux. The influence of the sample frequency and the influence of the signal frequency are ascertained with the two additional tests. The input parameter values for the three test cases are given in Table 3.1. The second test differs from the first only in its sample frequency and the third test differs from the first only in its blade passing

frequency. The values chosen are representative of a typical Kapton gauge used to measure the heat-flux in a turbine tested within a short duration facility.

Table 3.1 -- Steady Harmonic Test Case Input Parameters

$(\rho C_p k)_{gauge}$	330,625.	$(W^2s)/(m^4K^2)$	
$(\rho C_p k)_{blade}$	69,334,420.	$(W^2s)/(m^4K^2)$	
k/d	8,486	$W/(m^2K)$	
$\overline{Q}$	851,250	W/m^2	
$\overline{T}(0)$	400	K	
$\overline{T}(d)$	300	K	
Q_0	681,000	W/m^2	
ω	50,265 1/s	(blade pass freq. of 8,000 Hz), tests 1 & 2	25,132
	1/s	(blade pass freq. of 4,000 Hz), test 3	
f_s	100	kHz tests 1 & 3	
	200	kHz test 2	

Figure 3.1 gives the analytical upper and lower surface temperatures input into ACQ for test number 1. Figure 3.2 is a plot of both the theoretical heat-flux and the calculated ACQ heat-flux versus time for test number 1. This plot shows that the ACQ heat-flux lags the theoretical by approximately 0.005 ms which is one half a sample period. However, Figure 3.2 also shows that the calculated heat-flux is nearly identical to the theoretical heat-flux in every other respect; i. e., frequency, perturbation magnitude and mean.

The phase lag in the ACQ calculations was first investigated under the assumption that it was a programming error. However, no error could be identified. Closer examination suggested that it might be a numerical inaccuracy. The problem can be visualized by considering the theoretically phase lag between the temperature and heat-flux given by ϕ in Equation 3.3. It appears that this phase lag is not being accurately duplicated by ACQ. If the problem is numerical, the error should be reduced by a decrease in the integration step size. This hypothesis is investigated by test number 2.

Doubling the sample frequency in test 2 reduces the integration step size by one half. The result for this change is given in Figure 3.3 as a plot of the theoretical and calculated heat-flux. The lag between the calculated and theoretical heat-flux is approximately half that in test 1. As in test 1, the temporal lag is approximately half a time step.

From a practical stand point of reducing turbine data, the phase error is inconsequential as long as all the frequencies in the signal are shifted by the same amount. To ensure that this is the case, test 3 was conducted which has half the blade passing frequency of test 1. Figure 3.4 is a plot of the theoretical and calculated heat-flux for test number 3. These results show the same phase lag as test 1 indicating that the lag is not a function of the signal frequency.

To determine the accuracy of the steady harmonic heat-flux calculations, the theoretical heat-flux was shifted half a time step into phase with the calculated heat-flux to determine a meaningful difference between the two. Figure 3.5 gives the phase shifted theoretical heat-flux and the calculated heat-flux along with the difference between the two for test number 1. The plot shows that the calculated heat-flux is larger than theory while the function is increasing and lower while it is decreasing. The largest errors occur just after the min. and max. inflections. Figures 3.6 and 3.7 give the absolute and percent differences respectively for test number 1. Several points due exceed the 1% MIT stated accuracy in this test case.

3.1.2 -- Step in Surface Heat-flux

In the step change in surface heat-flux case both the transient and steady state solutions to the problem are of interest. The initial condition is

$$T(x,0) = T_i \tag{3.4}$$

For the surface heat-flux,

$$\dot{q} = -k\frac{\partial T}{\partial x}\bigg|_{x=0} = \begin{cases} 0, -\infty < t < t_0 \\ Q_0, t_0 \le t < \infty \end{cases} \tag{3.5}$$

the temperature distribution through the substrate is given by

$$T(x,t) = T_i + \frac{Q_0 d}{k}\left(\frac{t}{\tau}\right)\sum_{n=0}^{\infty}(-\Gamma)^n\left[\text{ierfc}(Y_n) - \Gamma\,\text{ierfc}(Z_n)\right] \tag{3.6}$$

where,

$$Y_n = \frac{x/d}{\sqrt{t/\tau}} + \frac{2n}{\sqrt{t/\tau}}$$

$$Z_n = -\frac{x/d}{\sqrt{t/\tau}} + \frac{2(n+1)}{\sqrt{t/\tau}}$$

and ierfc( ) is the first integral of the complementary error function. A gauge time constant τ is introduced based upon the gauge thermal diffusivity and substrate thickness,

$$\tau = \frac{1}{4}\left(\rho C_p k\right)\left(\frac{d}{k}\right)^2 \tag{3.7}$$

The input parameter values for this test case are given in Table 3.2. Again they are representative of a typical Kapton gauge.

Table 3.2 -- Input Parameters, Step Change in Surface Heat-flux

$(\rho C_p k)_{gauge}$	330,625.	$(W^2s)/(m^4K^2)$
$(\rho C_p k)_{blade}$	69,334,420.	$(W^2s)/(m^4K^2)$
k/d	8,486	$W/(m^2K)$
Q0	851,250	W/m^2
t0	5.0	ms
T_i	300	K
f_S	100	kHz

The analytical upper and lower surface temperatures, given by Equation 3.6, are shown in Figure 3.8 for the parameters of Table 3.2. These temperatures produce the calculated step in heat-flux shown in Figure 3.9. Figure 3.10 better shows the initial over shoot in the calculated heat-flux. The absolute and percent differences between the calculated and theoretical heat-flux steps are given in Figures 3.11 and 3.12 respectively.

These results show that the calculations recover in less than 0.3 ms (30 time steps) to match the magnitude of the step almost exactly. The over shoot in the calculated heat-flux is a numerical error due to the upper surface temperature's abrupt change in slope and the very large initial value of that slope. The 4% error of the over shoot is undesirable but acceptable considering its short duration along with the fact that a step change is a worse case example.

3.1.3 -- Convective Heat Transfer at the Surface of a Semi-infinite Slab

This test case examines the heat transfer at the surface of a semi-infinite slab given the initial condition

$$T(x,0) = T_i \tag{3.8}$$

and boundary conditions

Interior:
$$T(\infty,t) = T_i \tag{3.9}$$

Surface:
$$-k\left.\frac{\partial T}{\partial x}\right|_{x=0} = h\left[T_\infty - T(0,t)\right] \tag{3.10}$$

where h is the convective heat transfer coefficient and T_∞ is the free stream fluid temperature. The closed form solution for the temperature distribution in the slab as given by Incropera and DeWitt [1985]

$$\frac{T(x,t)-T_i}{T_\infty - T_i} = \operatorname{erfc}\left(\frac{x}{2\sqrt{t}}\sqrt{\frac{\rho C_p}{k}}\right) - \left[\exp\left(\frac{hx}{k}+\frac{h^2 t}{\rho C_p k}\right)\right]\left[\operatorname{erfc}\left(\frac{x}{2\sqrt{t}}\sqrt{\frac{\rho C_p}{k}}+h\sqrt{\frac{t}{\rho C_p k}}\right)\right] \tag{3.11}$$

The input parameter values for this test case are given in Table 3.3. These values are representative of a Pyrex-type gauge.

Table 3.3 -- Input Parameters, Semi-infinite Slab (Pyrex Insert Gauge)

$(\rho C_p k)_{gauge}$	2.84×10^6	$(W^2 s)/(m^4 K^2)$
k/d	551	$W/(m^2 K)$
d	0.00254	m
T_i	294	K
T_∞	533	K
h	1260	$W/(m^2 K)$
f_s	20	kHz

The analytical upper surface temperature generated by Equation 3.11 is given in Figure 3.13. The heat-flux calculated with this temperature trace is given along with the theoretical heat-flux in Figure 3.14. The absolute and percent differences between the calculated and theoretical heat-flux are given in Figures 3.15 and 3.16 respectively. These results show an initial error of nearly 10% in the calculations which disappears in approximately 0.3 ms to settle at a value near 0.1%.

3.2 -- Effects of Signal Anomalies

It is necessary to understand how errors in the recorded signals propagate through the heat-flux calculations. Two typical sources of signal errors are studied here. The first source considered is analog to digital conversion (A/D) which introduces error because of the finite resolution of A/D systems. The second source is random noise which is super-imposed onto the desired signal.

3.2.1 -- A/D Resolution Error Propagation

The effects of finite A/D resolution are studied using the steady harmonic surface heat-flux case with the same parameters as in test number 1. The study simulates the DC coupled acquisition of the temperature signals which have small perturbations in comparison to their mean levels; reference Figure 3.1. Therefore, the range of perturbation temperatures is characterized by a small percentage of the A/D system's available bits. In the case presented here, 1.5% of 4096 available bits were used to characterize the +/- 3 K upper surface temperature perturbation about its 400 K mean value.

The calculated and theoretical heat-flux plots for this test case are shown in Figure 3.17. This figure appears to be identical to Figure 3.2, but the differences given in Figures 3.18 and 3.19 show the truncation errors caused by the low A/D resolution. The error for this case is as high as 5% of the mean heat-flux and 13.5% of the local heat-flux. This represents a factor of three increase in the error strictly due to the low A/D resolution.

The test demonstrated here is a worse case. This type of A/D truncation error can be greatly reduced by utilizing the A/D system's resolution to its fullest. For instance, a DC offset can be used to zero the smallest expected voltage reading. Then, a gain can be applied to the expected signal so that it will fill the full A/D range. The signal should be acquired in a DC couple mode because of the decaying nature of the signal in short duration test facilities. However, AC coupled acquisition can be performed over short (quasi steady) periods with the DSP data channels but is not recommended with the Data Lab channels because of their poor RC time constant.

3.2.2 -- Random Noise Error Propagation

The steady harmonic surface heat-flux test case is used to study how random signal noise affects the accuracy of the heat-flux calculations. Signal noise is simulated by a

random number generator which super-imposes an error onto the exact temperature signal. This random error is applied to the signal at the digital level in terms of plus or minus a number of bits. The noise has a maximum possible value which is specified as a percentage of the full A/D bit range.

For the case studied here, the A/D system is assumed to have a 4096 bit range. The exact temperatures from test number 1 are characterized with 1.5% of the available bits in the simulated DC couple mode. A 0.2% of full scale noise level is super-imposed onto the temperatures. This level of noise corresponds to approximately +/- 0.3 K for this particular test case. Figure 3.20 compares the exact upper temperature from Figure 3.1 to the simulated upper temperature with the A/D resolution error and random noise. The signals only differ slightly with the effect of the random noise being seen in the separation of some of the data points.

The ACQ heat-flux calculated from the simulated temperature data is given in Figure 3.21 along with the theoretical heat-flux. Again the phase shift is present in the calculations and the effect of the signal anomalies is discernible. Figures 3.22 and 3.23 give the absolute and percent differences which indicate the random nature of the error. While most of the calculations are within +/- 5%, there are a few that are over 20%.

This test, like the previous one, is a worse case scenario. For example, the A/D resolution is much lower than could be achieved, as explained in Section 3.2.1. Also, Epstein et al. [1985] states that electrical signal-to-noise considerations are not important and quotes an analysis performed by Guenette [1985] which shows that the noise equivalent temperature is only 0.002 K.

3.3 -- Actual Data Reduction

The last part of the code verification process entails the reduction of heat-flux gauge data from a turbine experiment. Data from a single sensor semi-infinite Pyrex gauge was chosen because it could be accurately reduced by an unrelated method developed by Cook and Felderman [1966]. The Cook/Felderman method is regarded as a tried and true method for reducing semi-infinite heat-flux gauge data. The heat-flux data was taken from an SSME turbo pump test conducted in Calspan's Turbine Test Facility (TTF). The recorded gauge surface temperature is given in Figure 3.24.

The heat-flux calculated by both ACQ and Cook/Felderman are given in Figure 3.25. The plot shows that the two methods are in excellent agreement. A more detailed plot over the usable test time is given by Figure 3.26. This plot shows that the ACQ calculations are consistently 2.0% to 2.5% lower than the Cook/Felderman calculations. Considering the data and the difference in the methods, the correlation is very encouraging.

3.4 -- Results Summary

- Steady harmonic surface heat-flux: +3.25% to -1.2% error
- Step change in surface heat-flux: +3.25% (initial over shoot for ~0.3 ms) to +0.05%
- Convective heat transfer over semi-infinite slab: +10% (initial error for ~0.3 ms) to +0.1% error
- A/D resolution error: +13.5% (most points below +5%) to-3.2% error
- A/D resolution plus 0.2% full scale random noise error: +28% to -22% error (most points between +/- 5%)
- Pyrex-type gauge data from SSME turbine test: -2.5% to -2.0% difference between Cook/Felderman

3.5 -- Conclusions

The ACQ heat-flux data reduction code has been verified. The code provides physically correct and accurate results when compared to various exact analytical solutions and one other accepted data reduction code. The propagation of signal error through the data reduction calculations has been investigated. Under what should be worse case data acquisition conditions, the simulated data errors produced significant but acceptable errors in the heat-flux calculations. The random errors encountered would be nearly eliminate with adequate ensemble averaging.

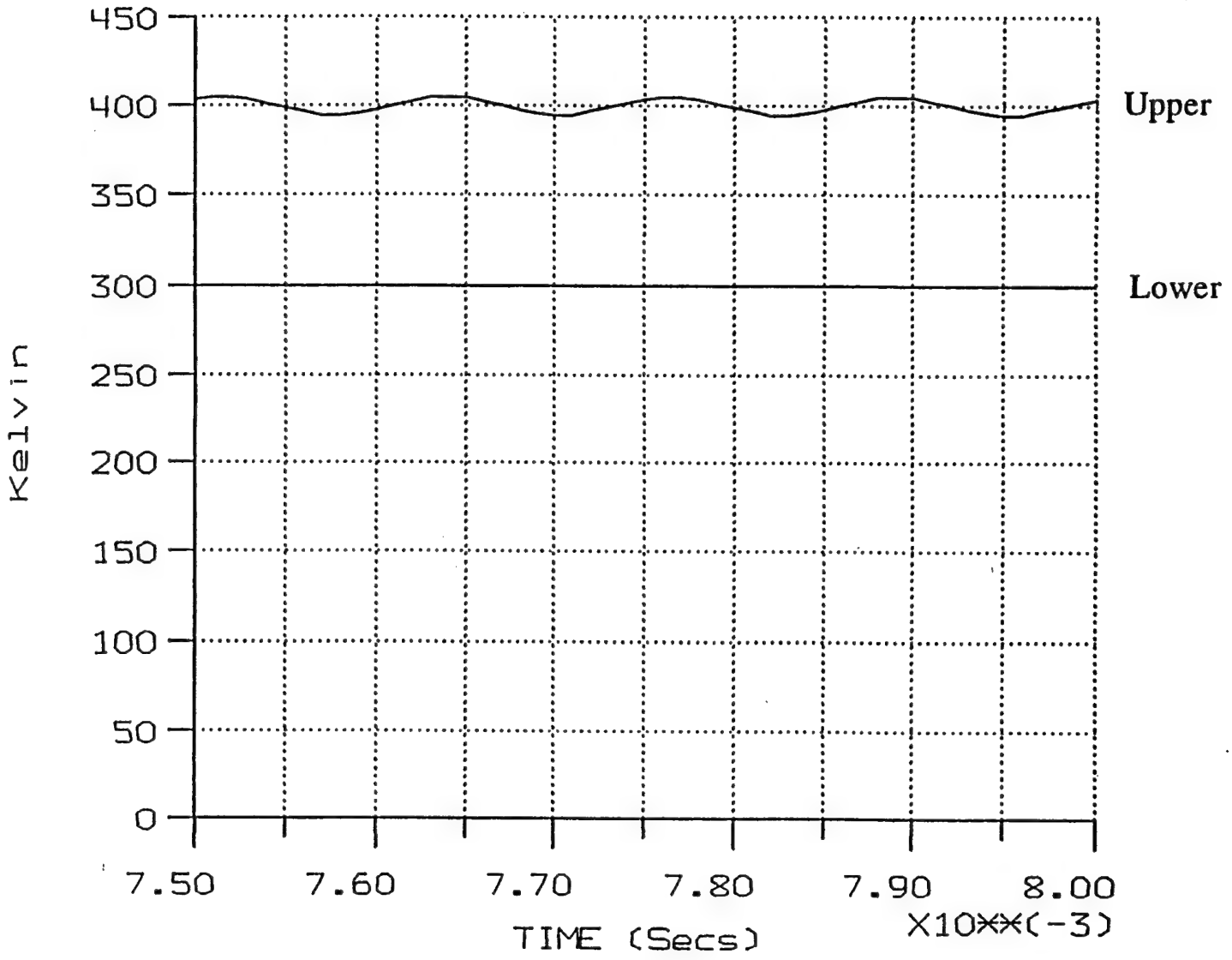


Figure 3.1 -- Exact analytical upper and lower surface temperatures for steady harmonic surface heat-flux

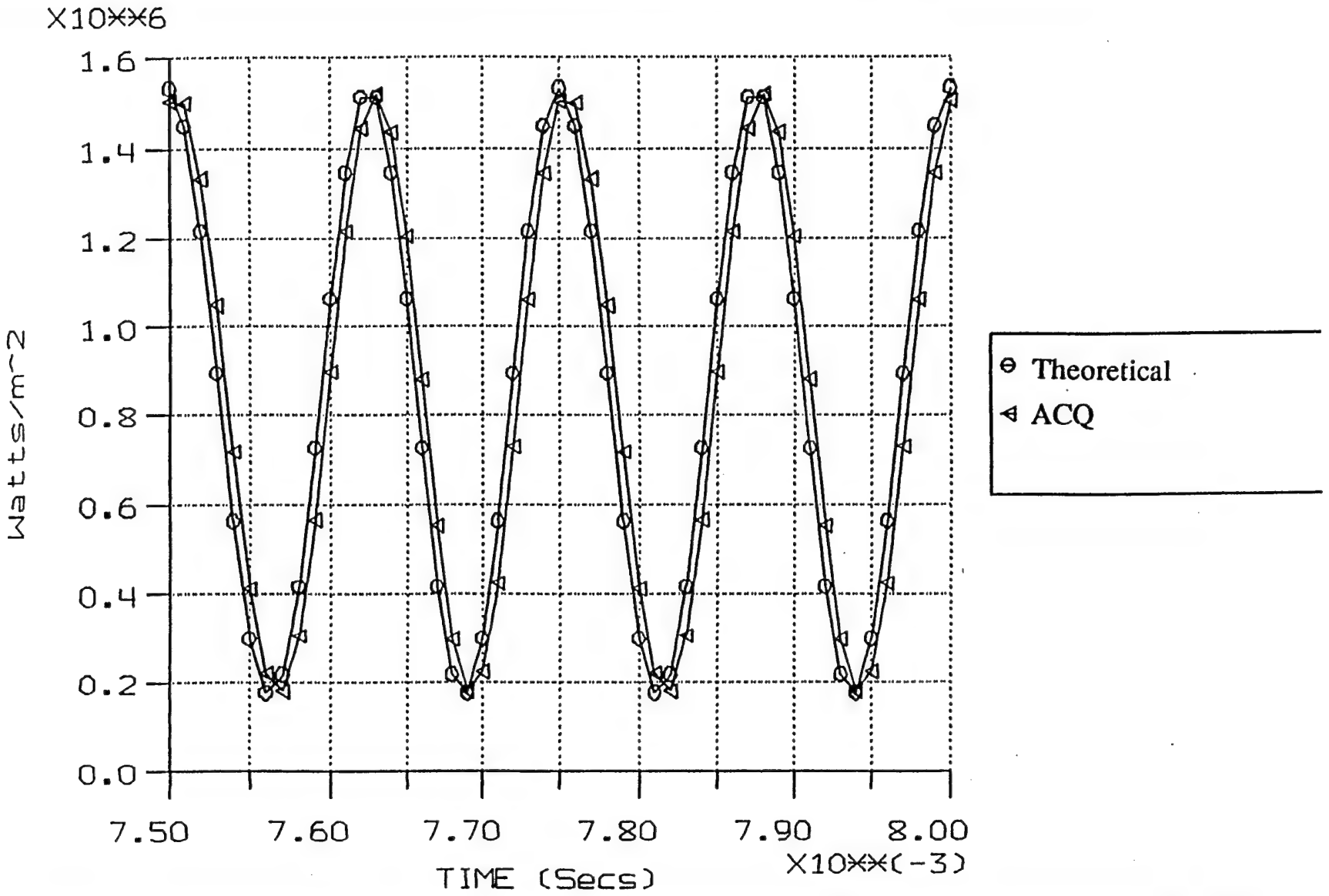


Figure 3.2 -- Theoretical and ACQ calculated steady harmonic surface heat-flux; 100 kHz sample frequency and 8 kHz blade passing frequency

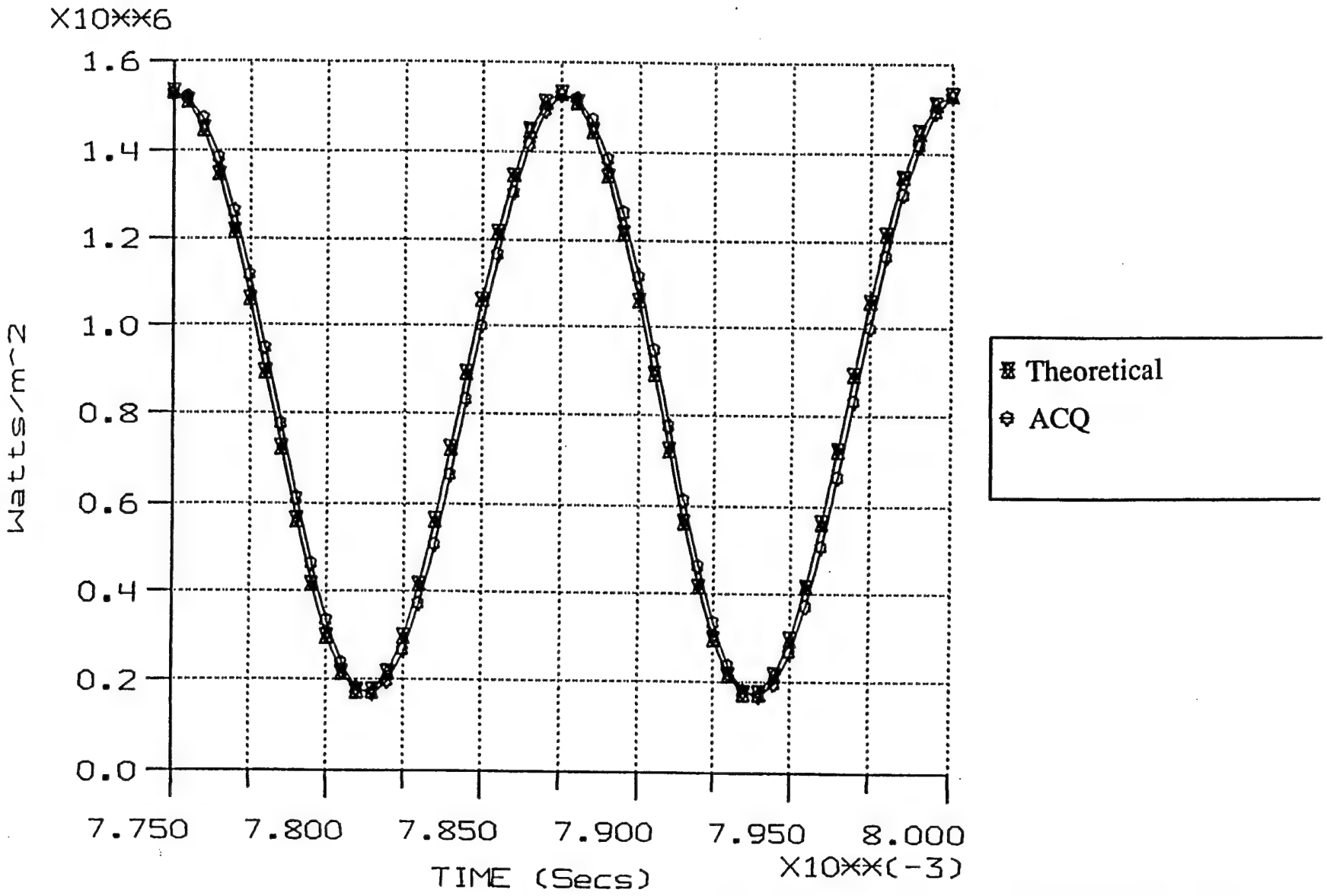


Figure 3.3 -- Theoretical and ACQ calculated steady harmonic surface heat-flux; 200 kHz sample frequency and 8 kHz blade passing frequency

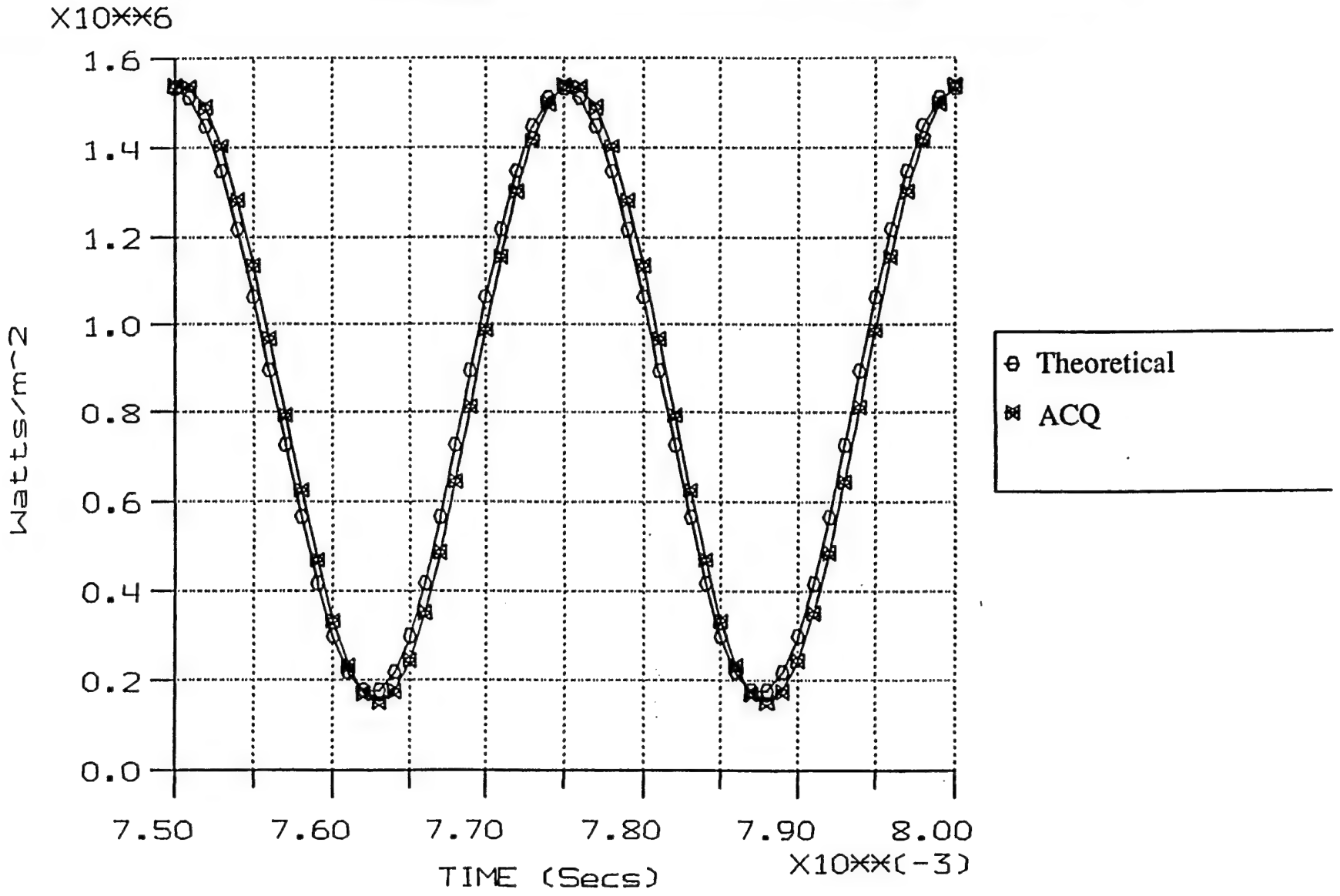


Figure 3.4 -- Theoretical and ACQ calculated steady harmonic surface heat-flux; 100 kHz sample frequency and 4 kHz blade passing frequency

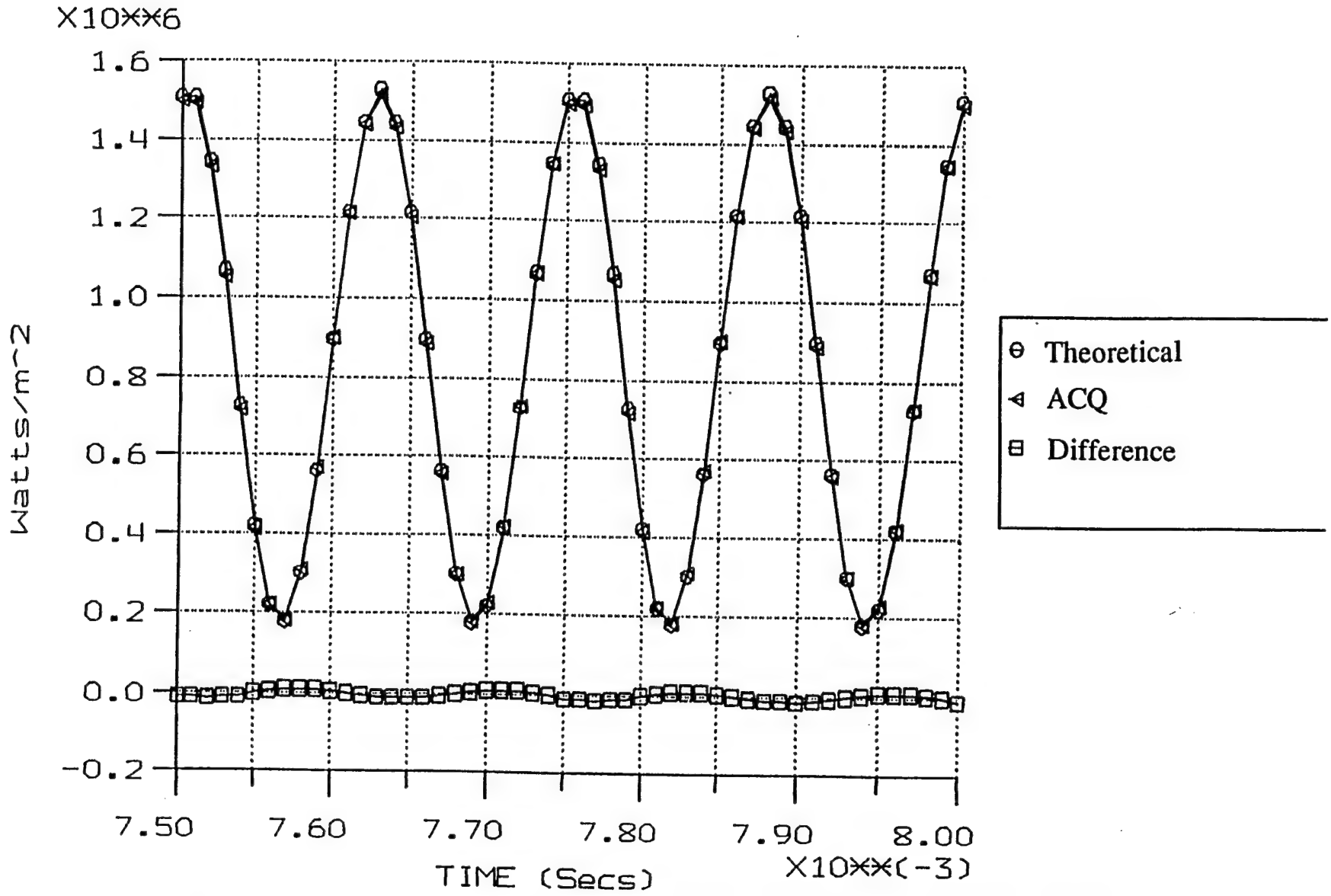


Figure 3.5 -- Phase shifted theoretical steady harmonic surface heat-flux and the ACQ calculated heat-flux

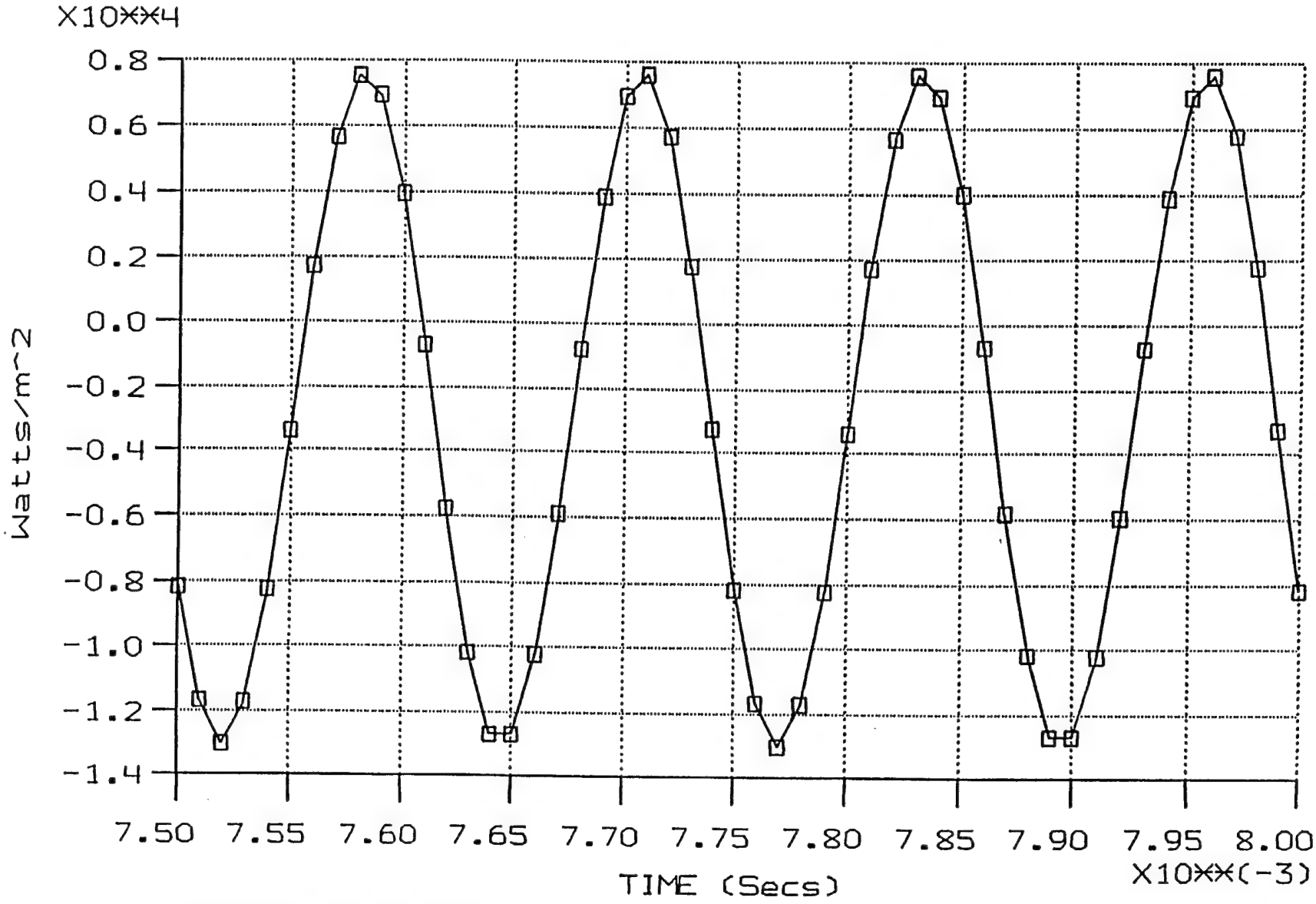


Figure 3.6 -- Absolute difference between theoretical and ACQ steady harmonic surface heat-flux

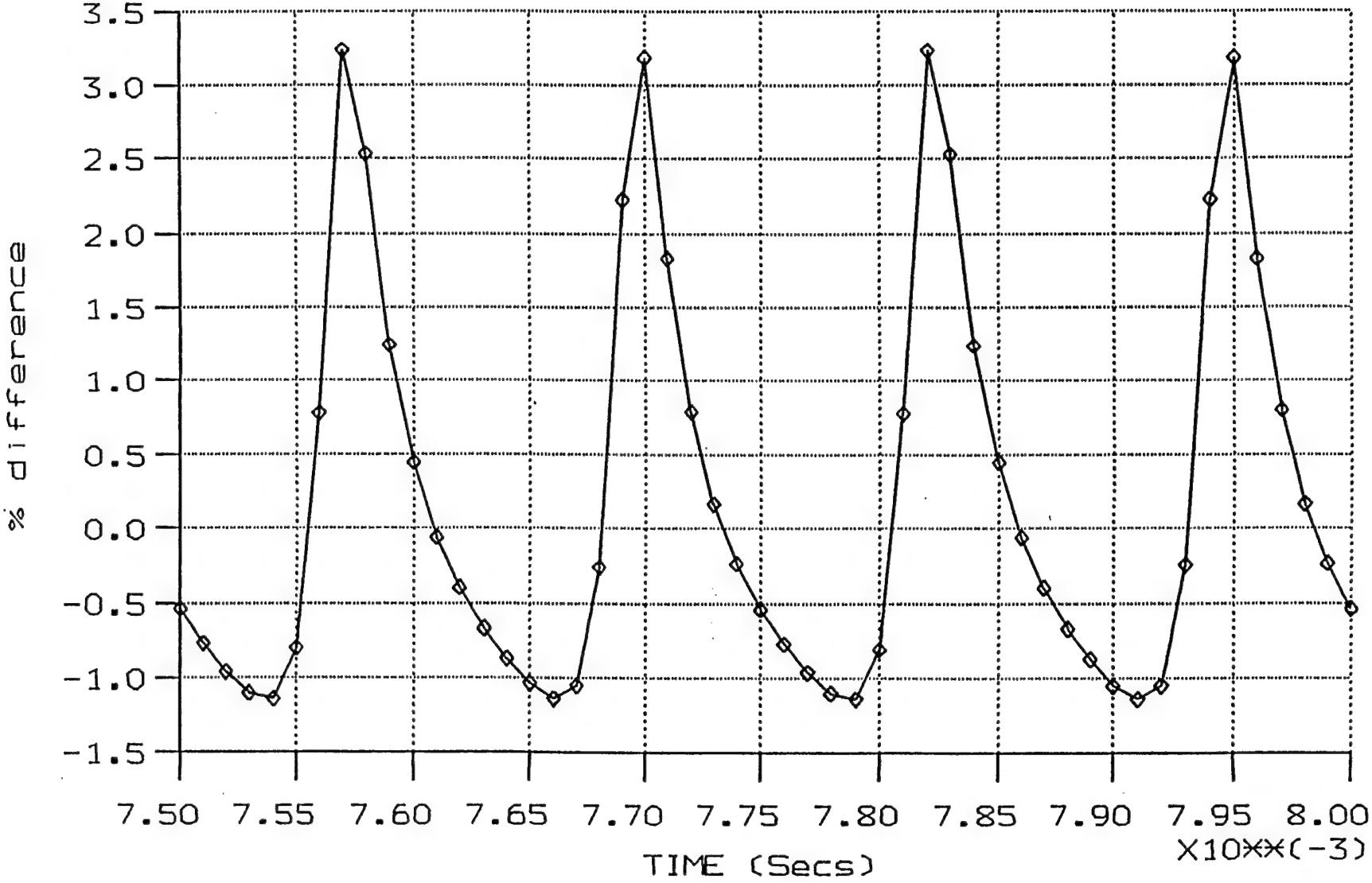


Figure 3.7 -- Percent difference between theoretical and ACQ steady harmonic surface heat-flux

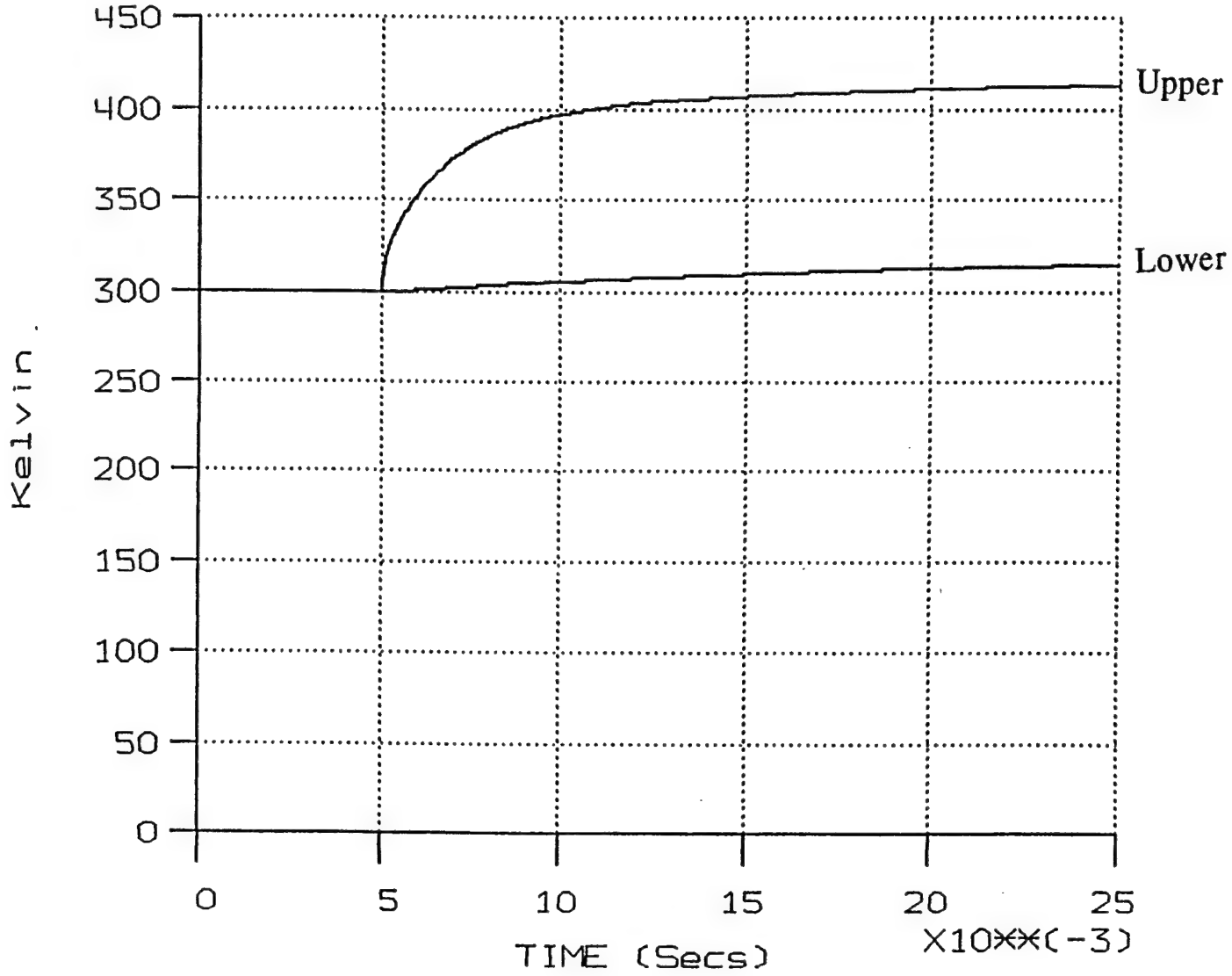


Figure 3.8 -- Exact analytical upper and lower surface temperatures for a step change in surface heat-flux

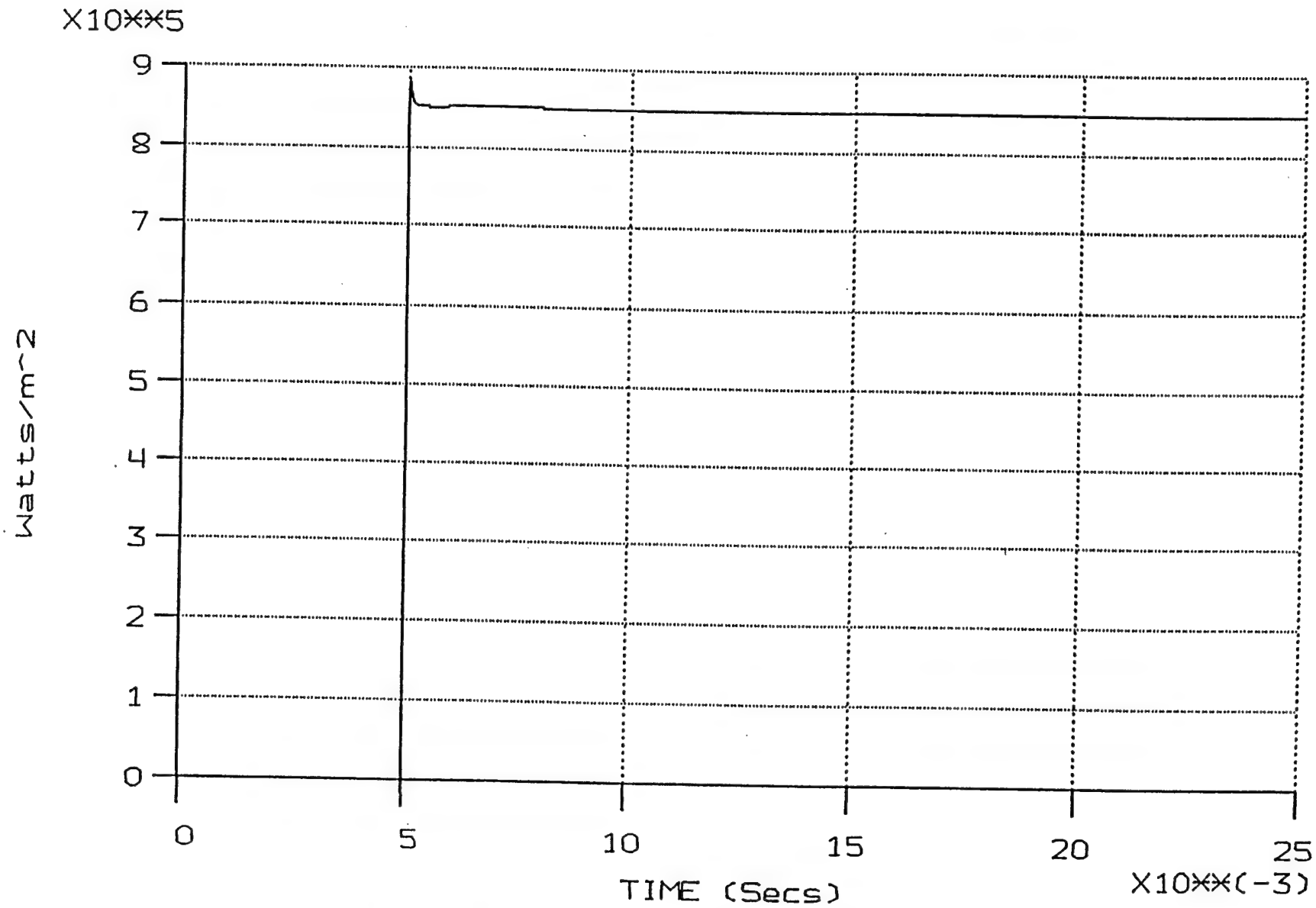


Figure 3.9 -- ACQ calculated step change in surface heat-flux

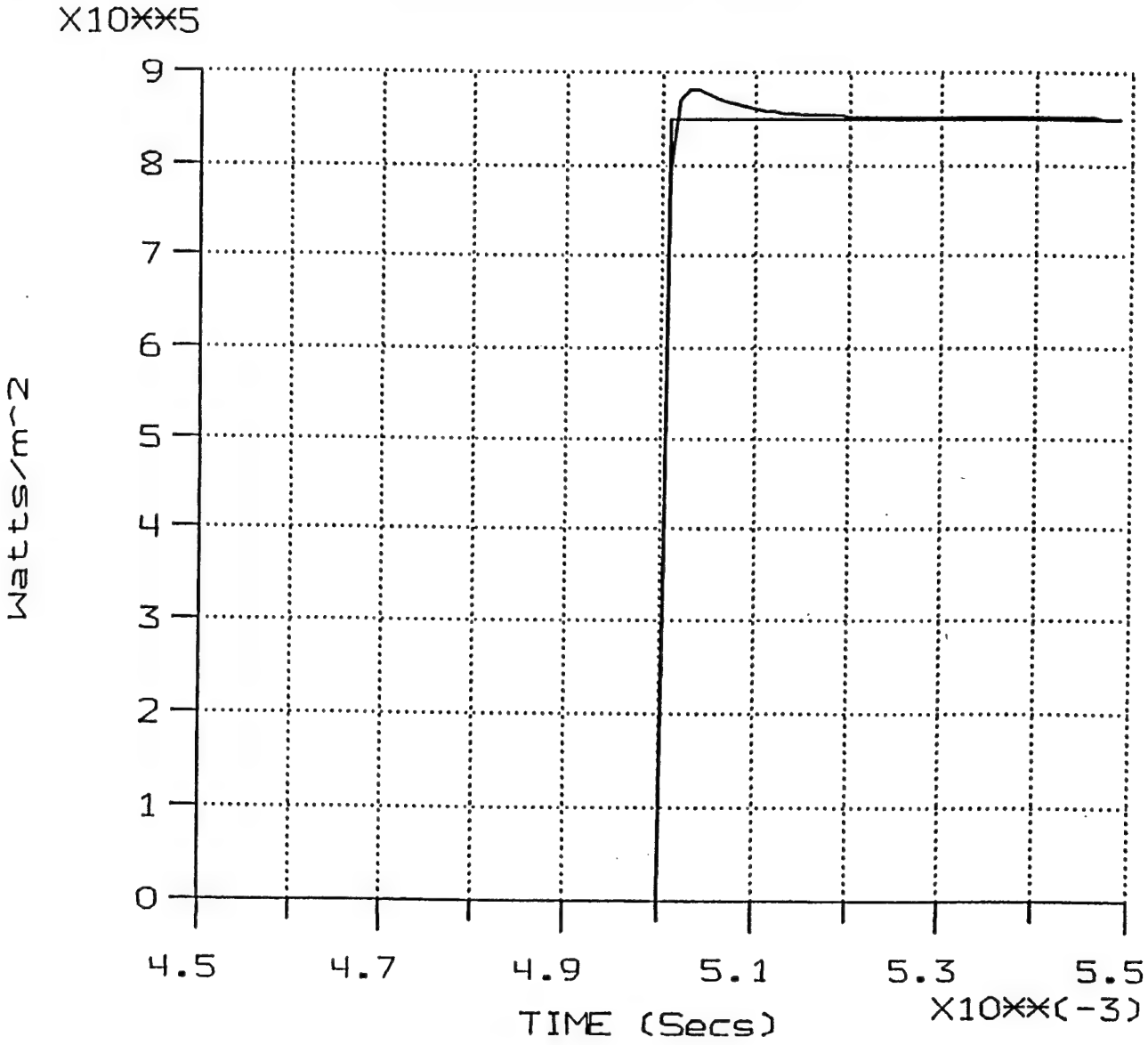


Figure 3.10 -- Theoretical and ACQ calculated step change in surface heat-flux

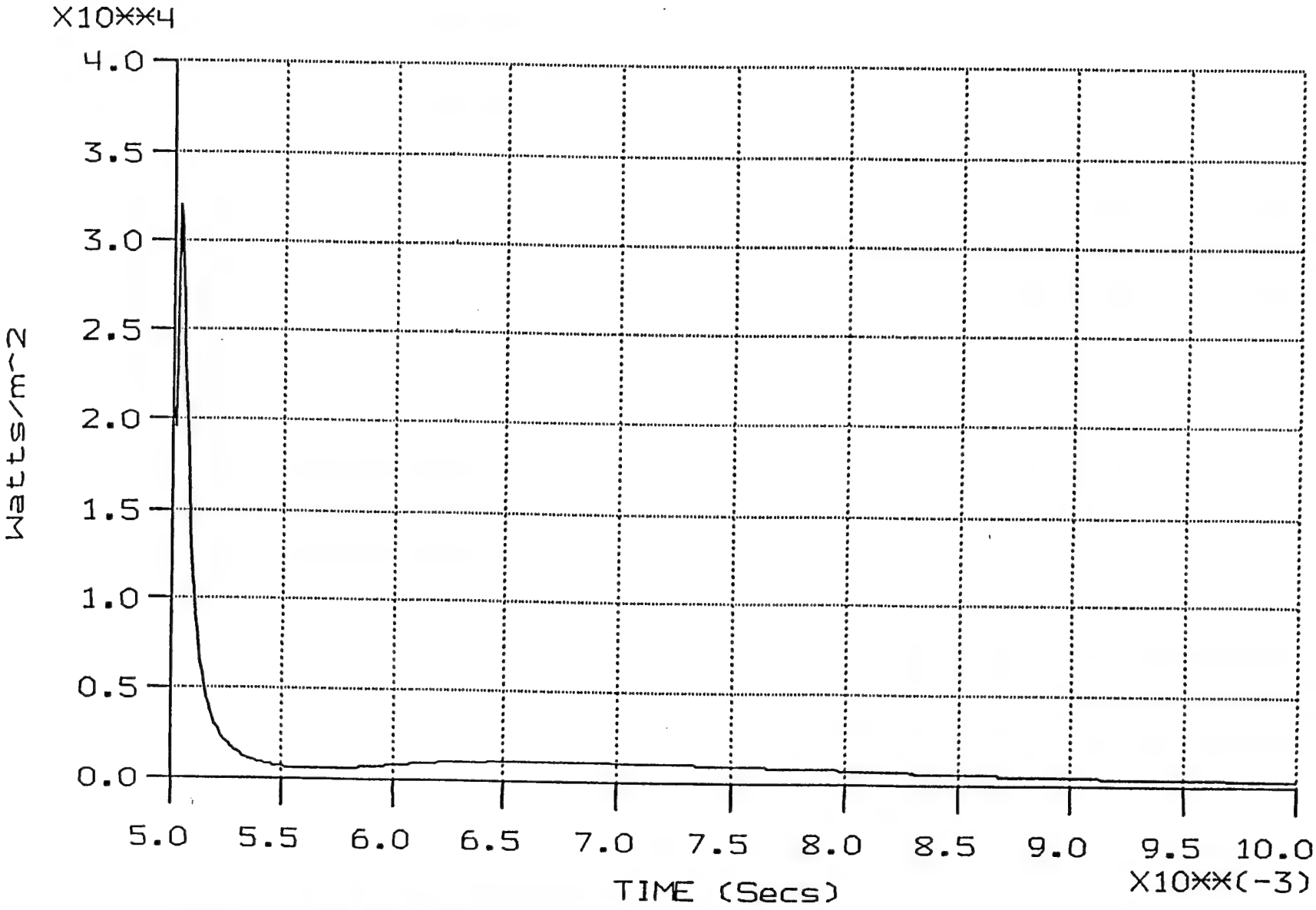


Figure 3.11 -- Absolute difference between theoretical and ACQ step change in surface heat-flux

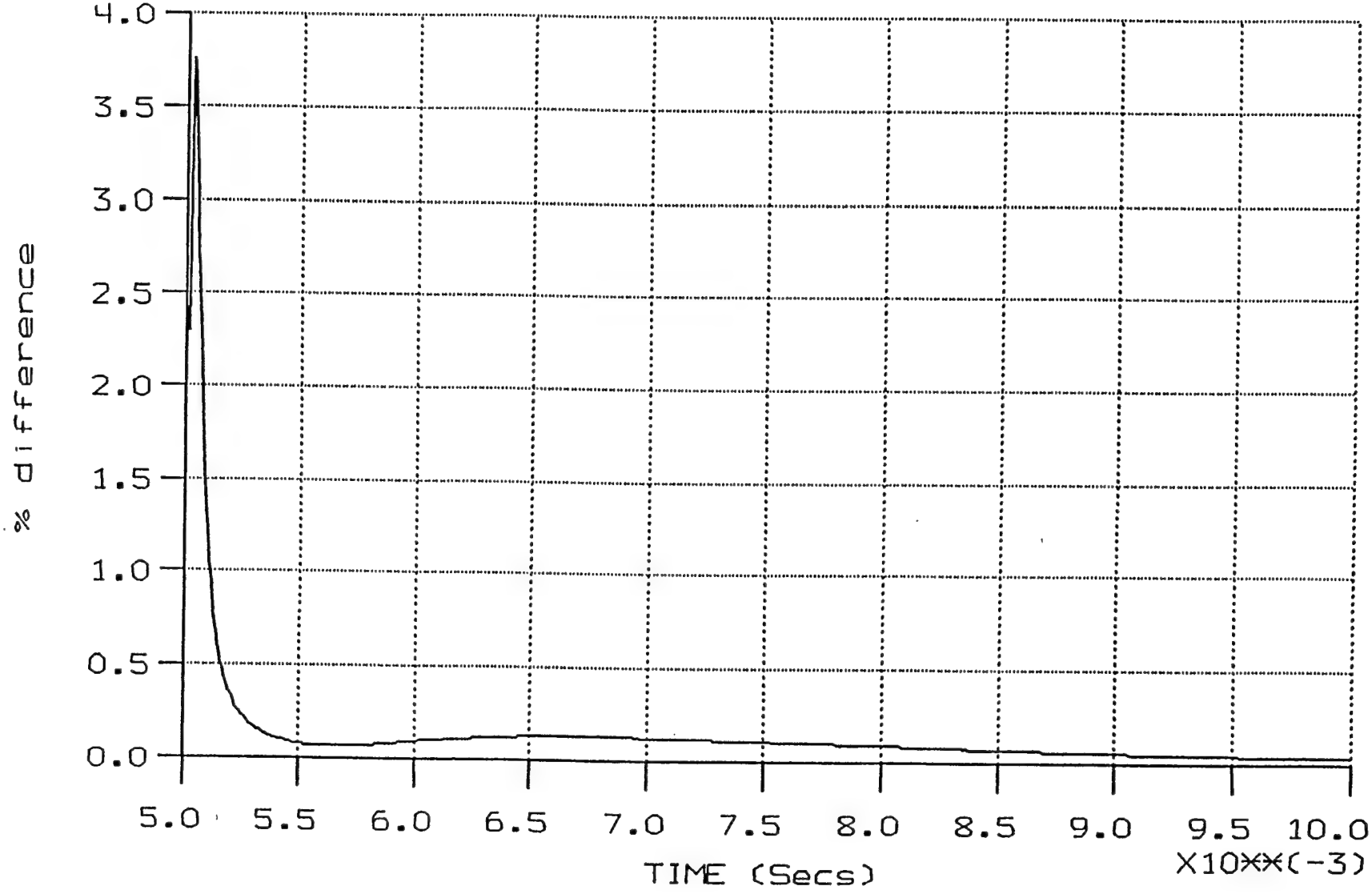


Figure 3.12 -- Percent difference between theoretical and ACQ step change in surface heat-flux

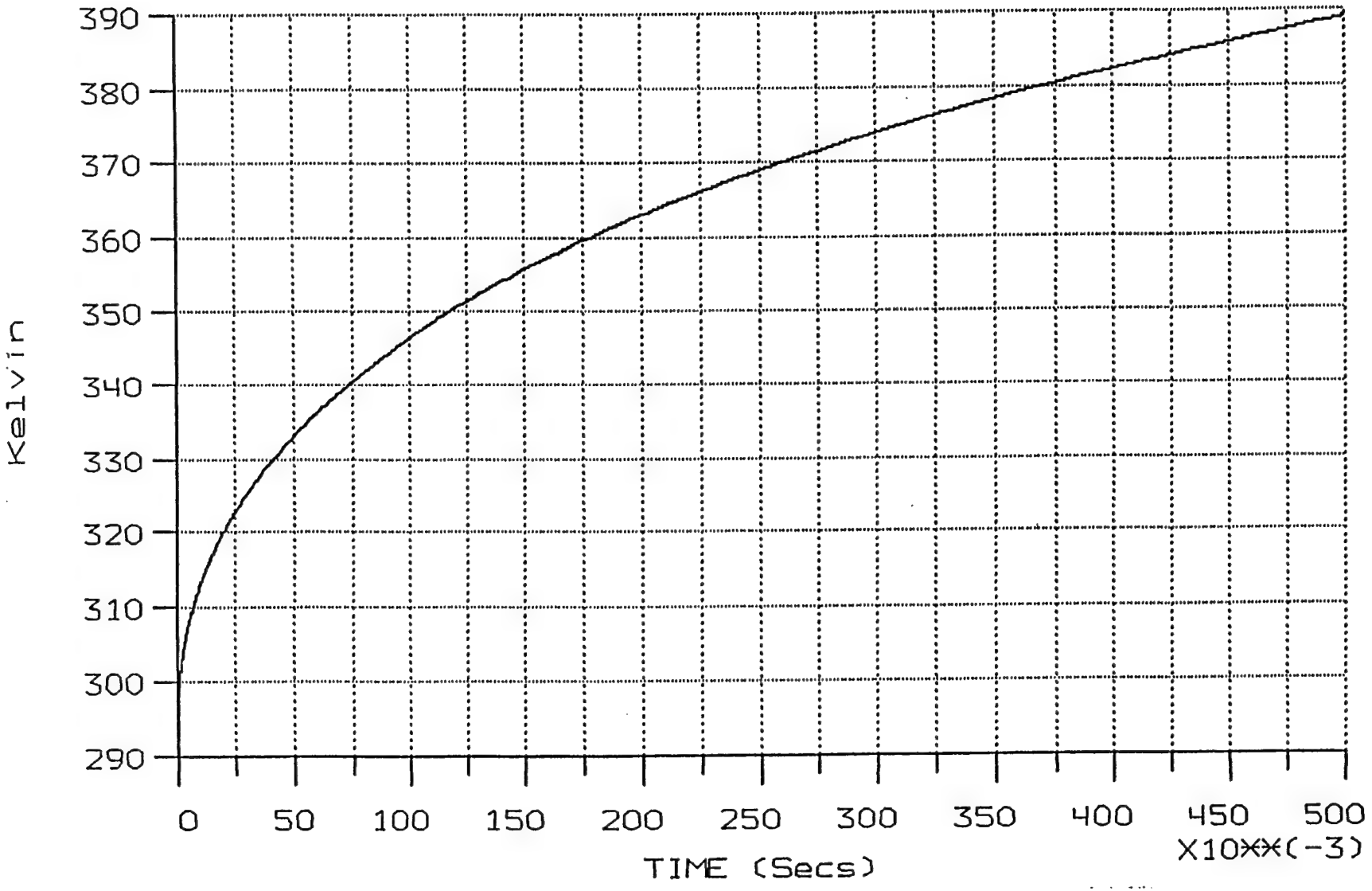


Figure 3.13 -- Exact analytical upper surface temperature for convection over a semi-infinite slab

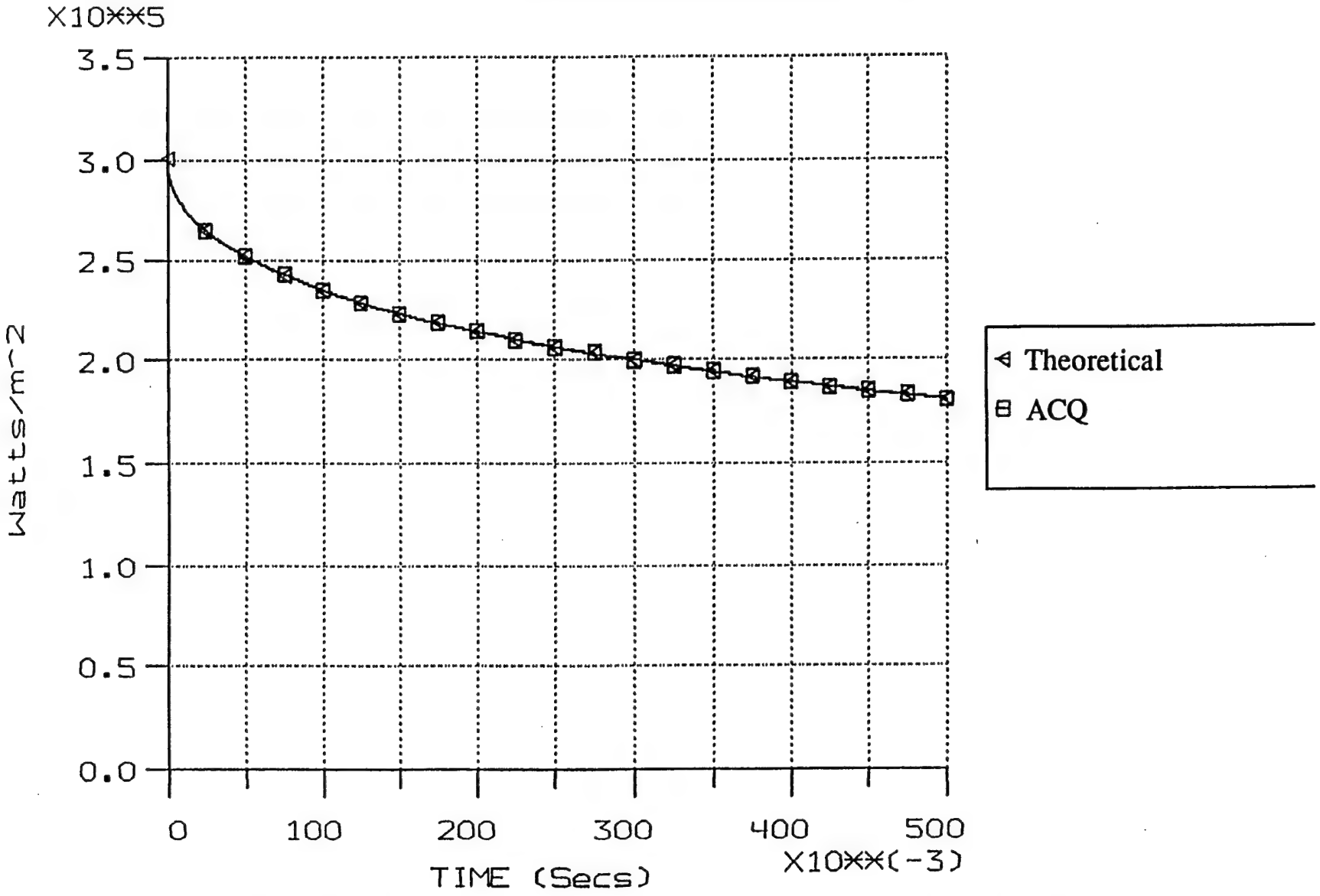


Figure 3.14 -- Theoretical and ACQ calculated heat-flux for a semi-infinite slab in convection

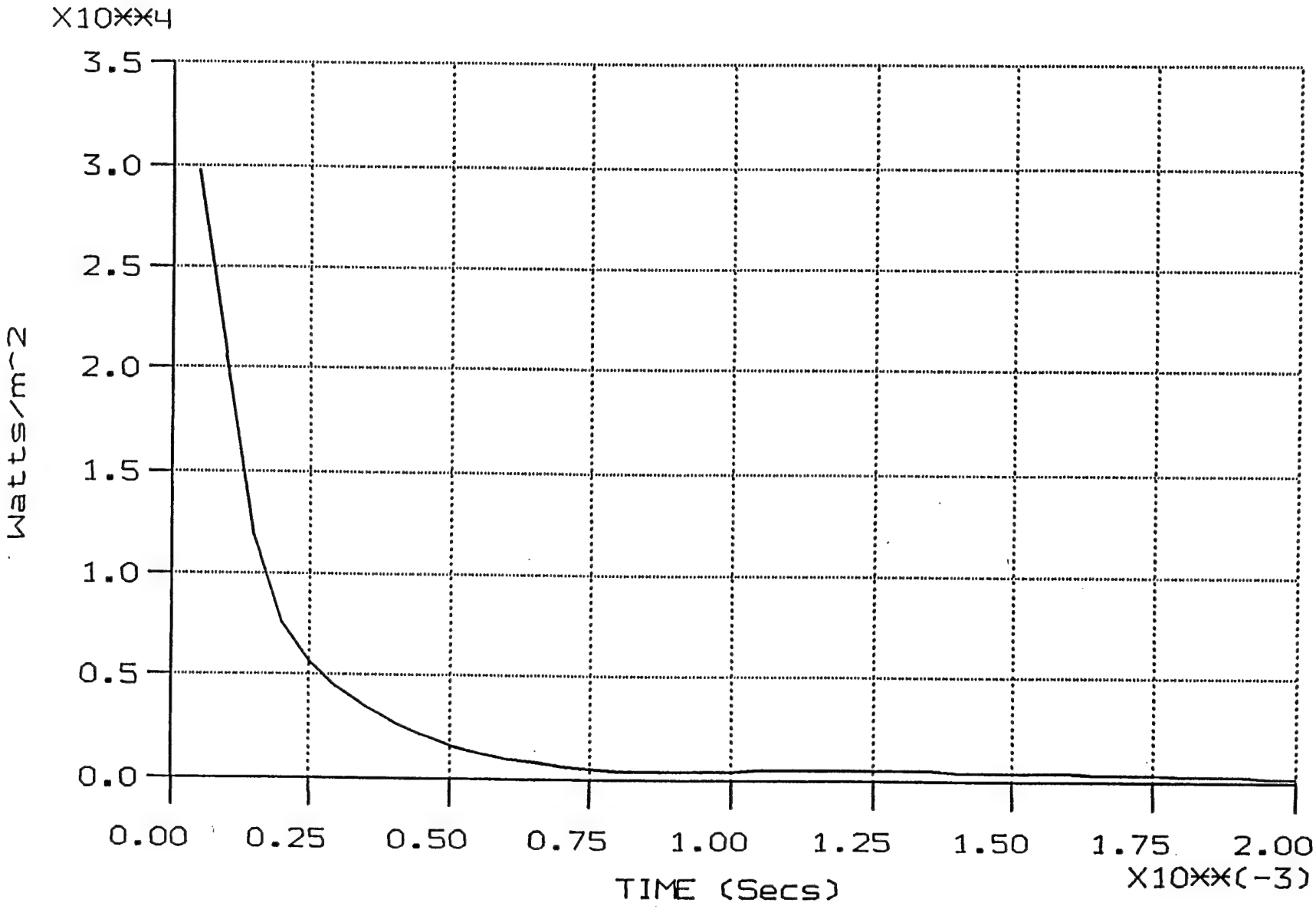


Figure 3.15 -- Absolute difference between theoretical and ACQ surface heat-flux for a semi-infinite slab in convection

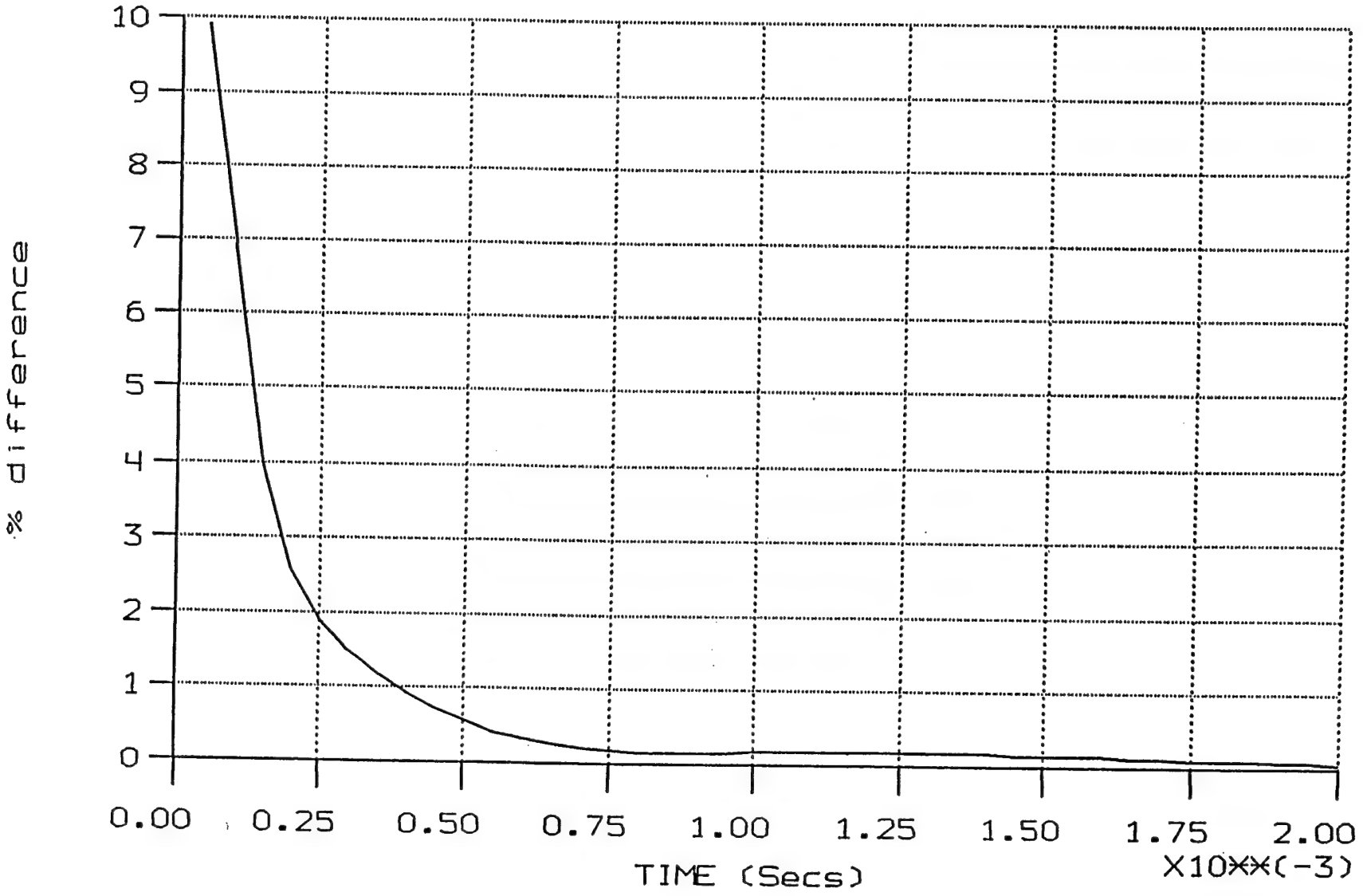


Figure 3.16 -- Percent difference between theoretical and ACQ surface heat-flux for a semi-infinite slab in convection

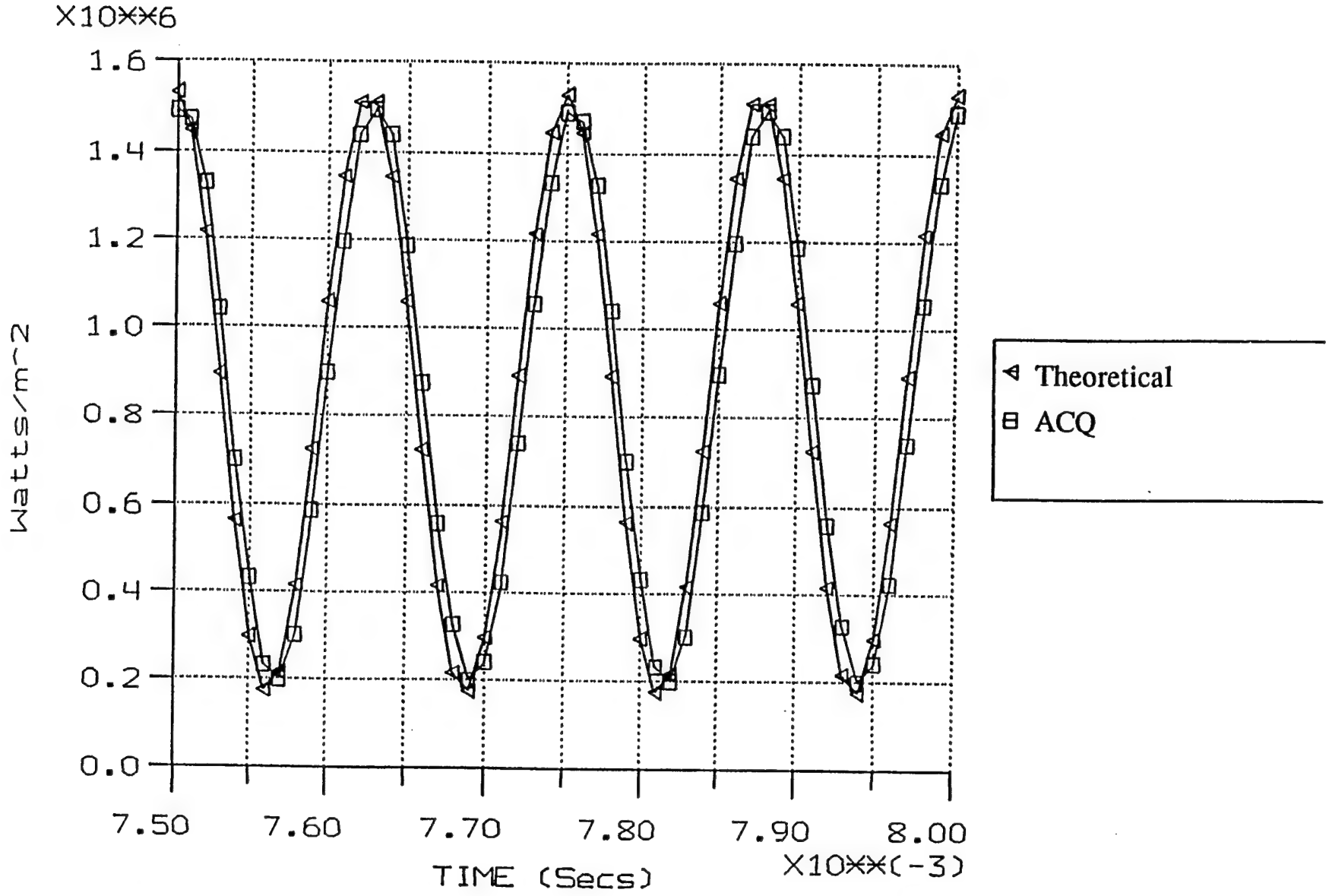


Figure 3.17 -- Theoretical and ACQ calculated steady harmonic heat-flux with 1.5% of the A/D resolution

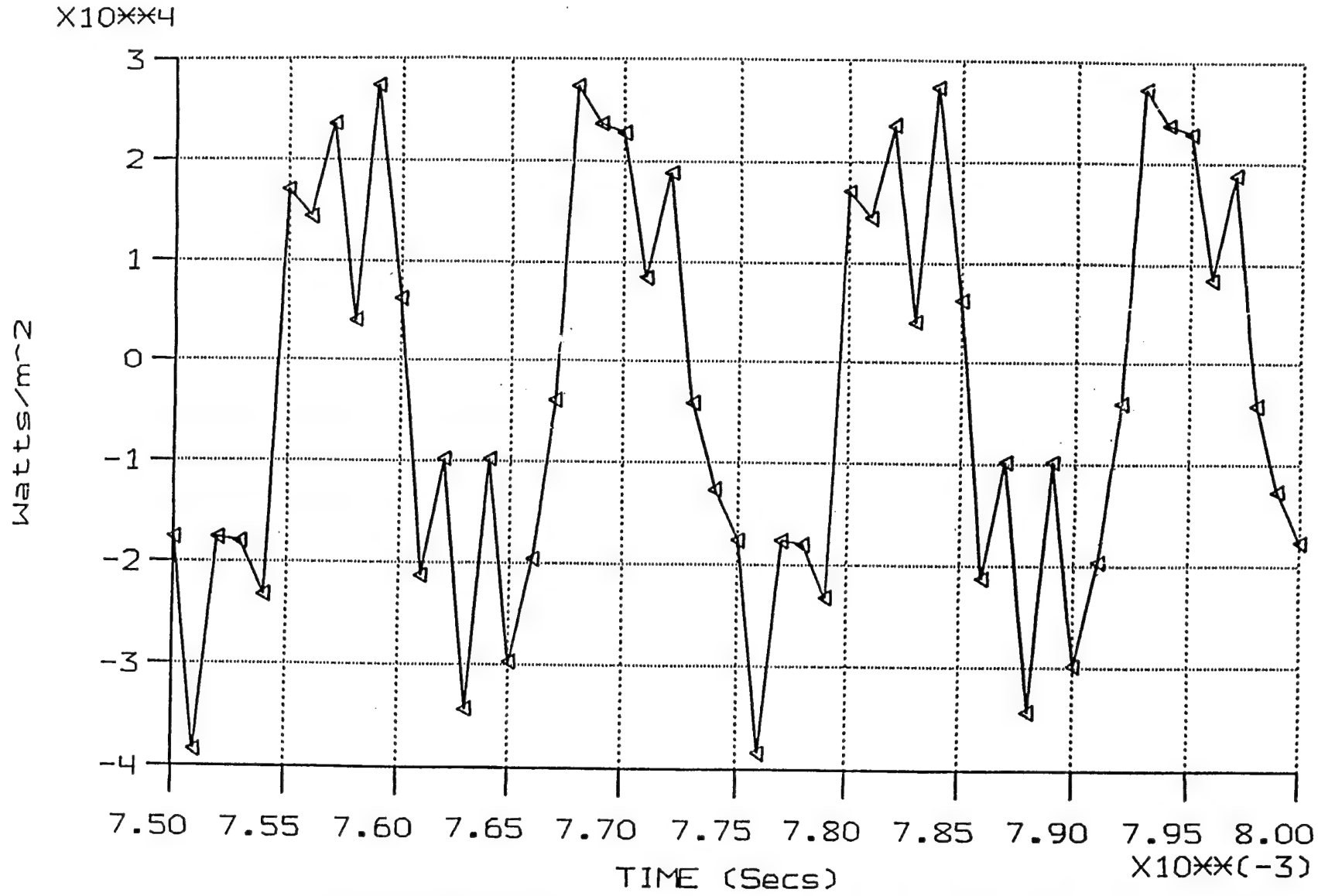


Figure 3.18 -- Absolute difference between theoretical and ACQ steady harmonic heat-flux with 1.5% of the A/D resolution

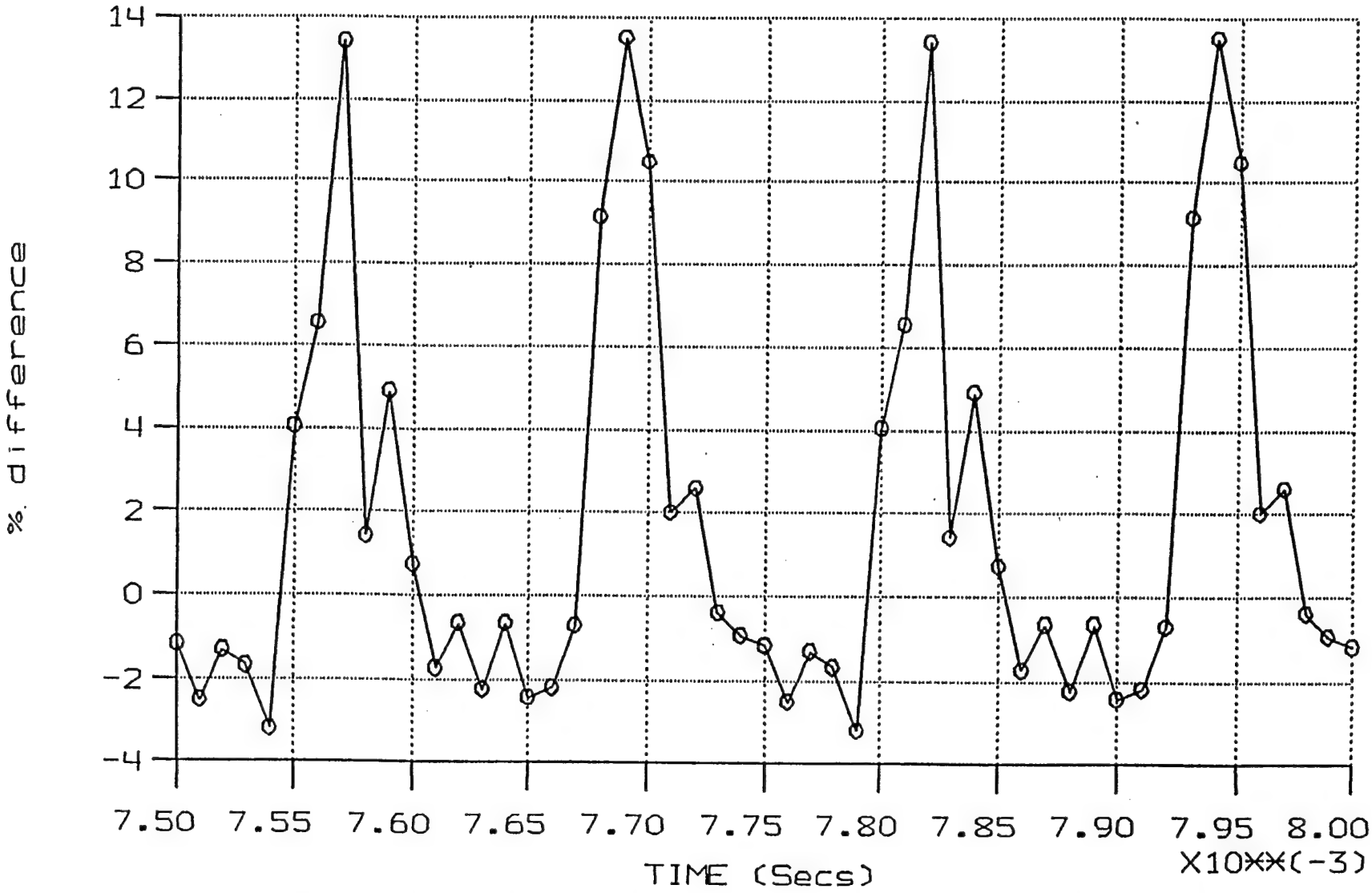


Figure 3.19 -- Percent difference between theoretical and ACQ steady harmonic heat-flux with 1.5% of the A/D resolution

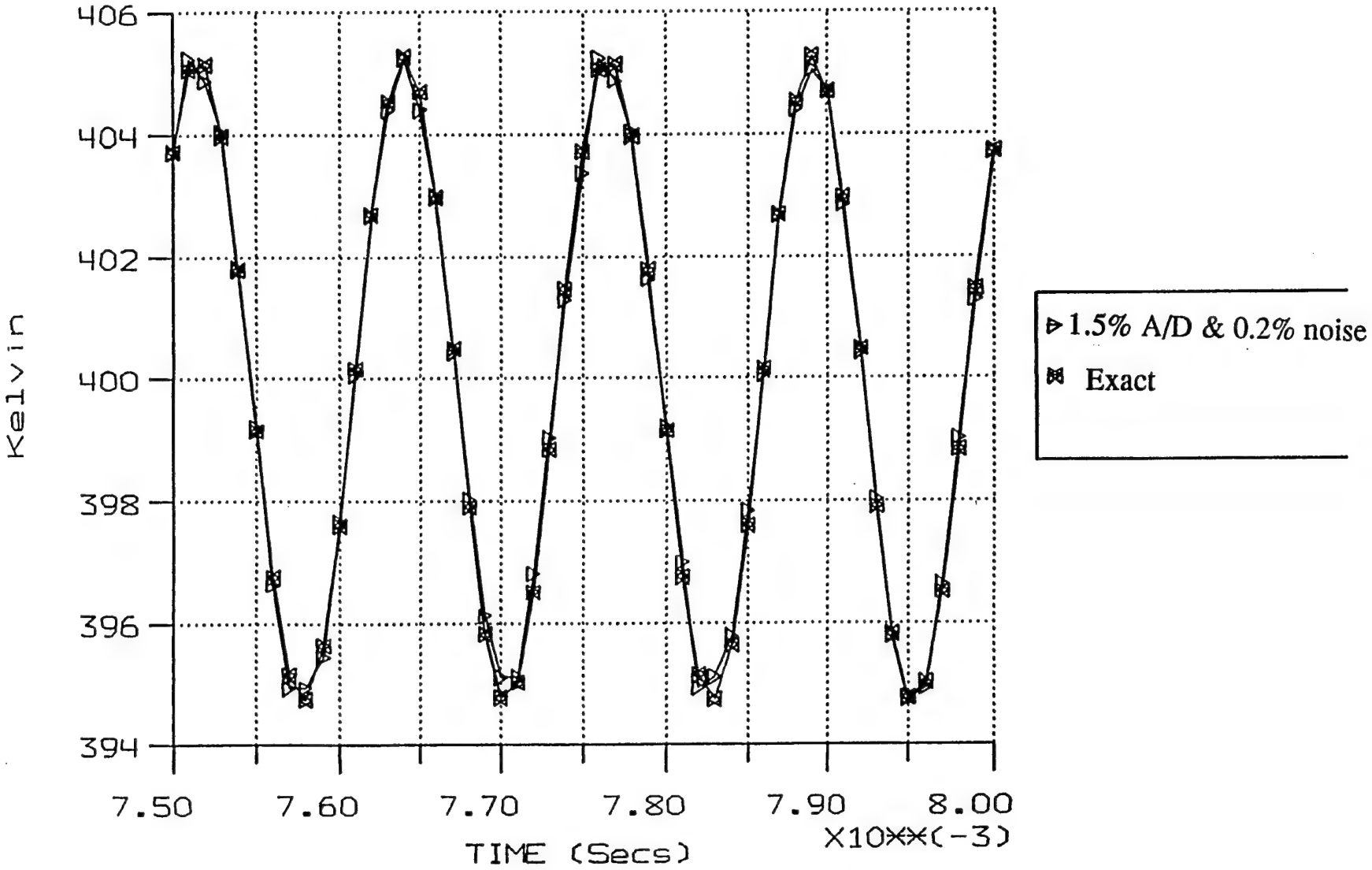


Figure 3.20 -- Exact analytical upper surface temperature with and without 1.5% A/D resolution and 0.2% random noise

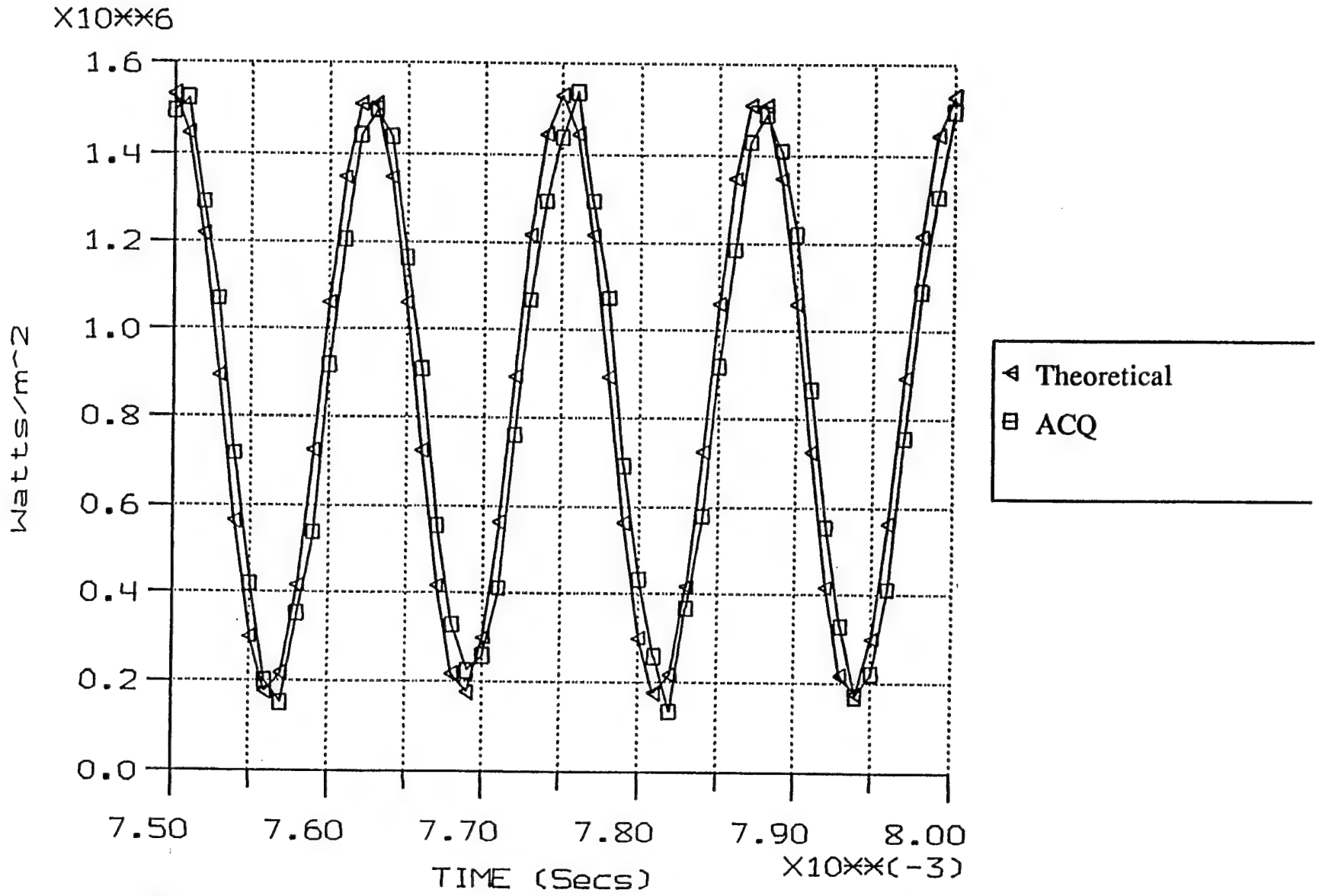


Figure 3.21 -- Theoretical and ACQ calculated steady harmonic surface heat-flux with 1.5% A/D resolution and 0.2% random noise

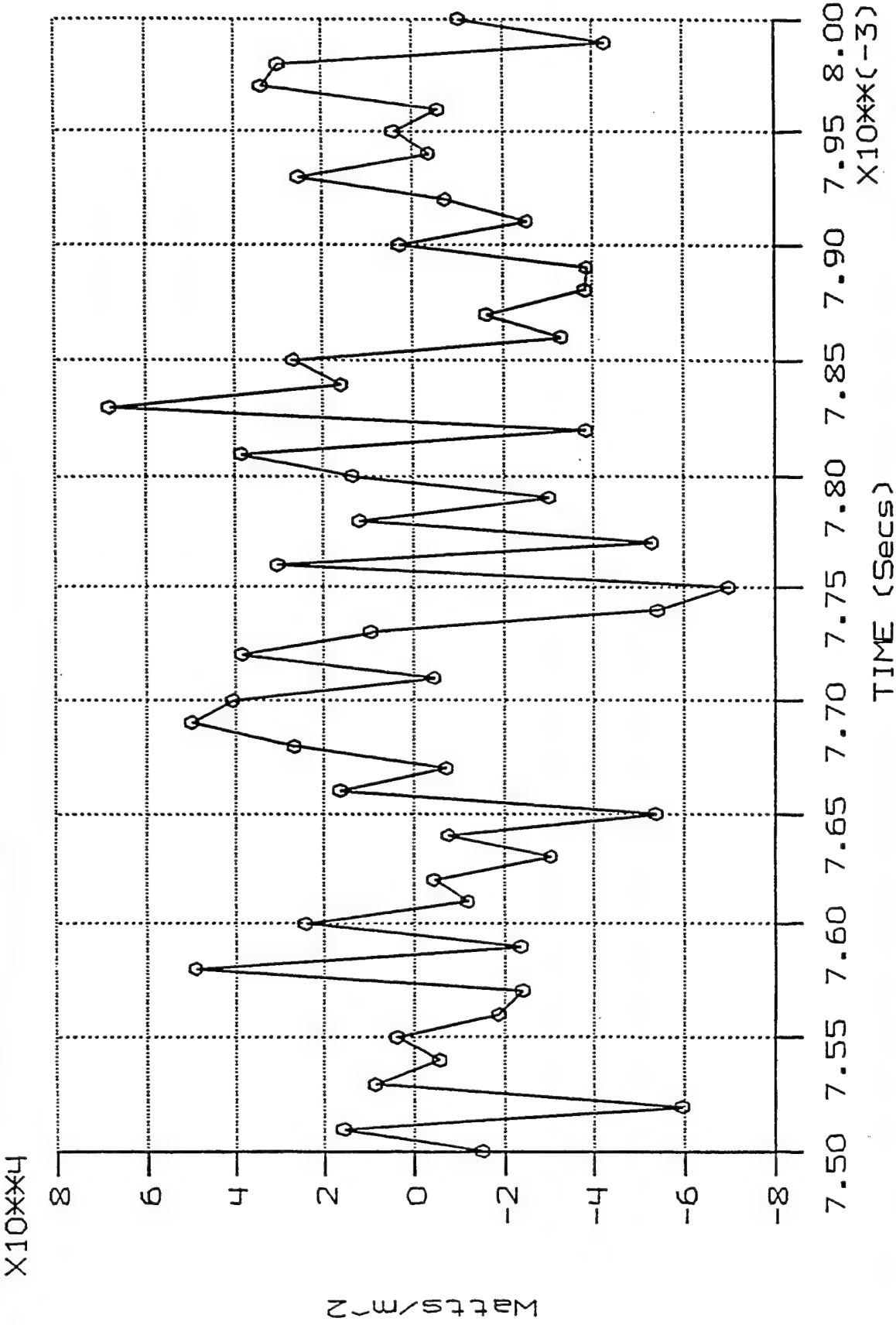


Figure 3.22 -- Absolute difference between theoretical and ACQ steady harmonic surface heat-flux with 1.5% A/D resolution and 0.2% random noise

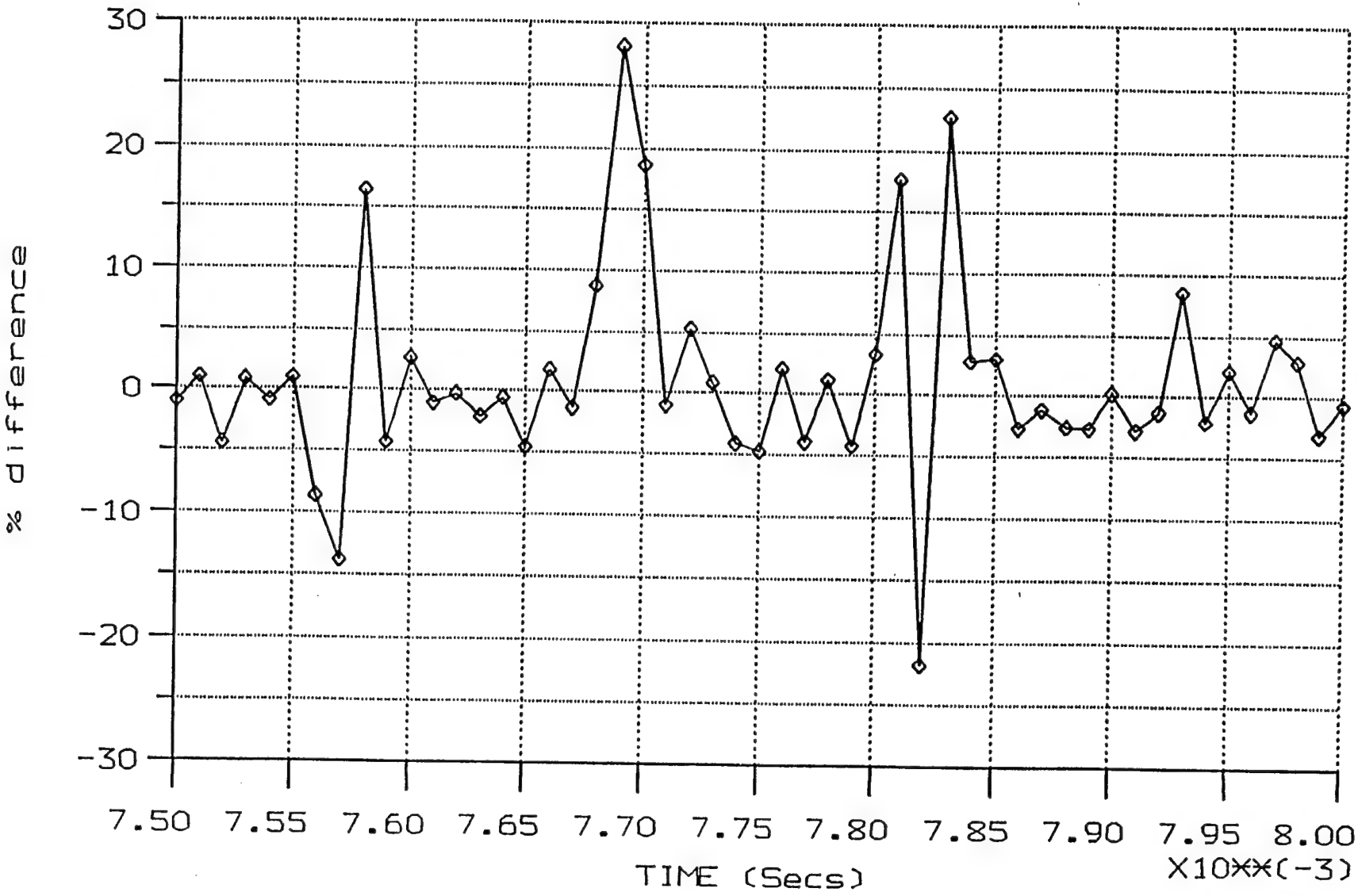


Figure 3.23 -- Percent difference between theoretical and ACQ steady harmonic surface heat-flux with 1.5% A/D resolution and 0.2% random noise

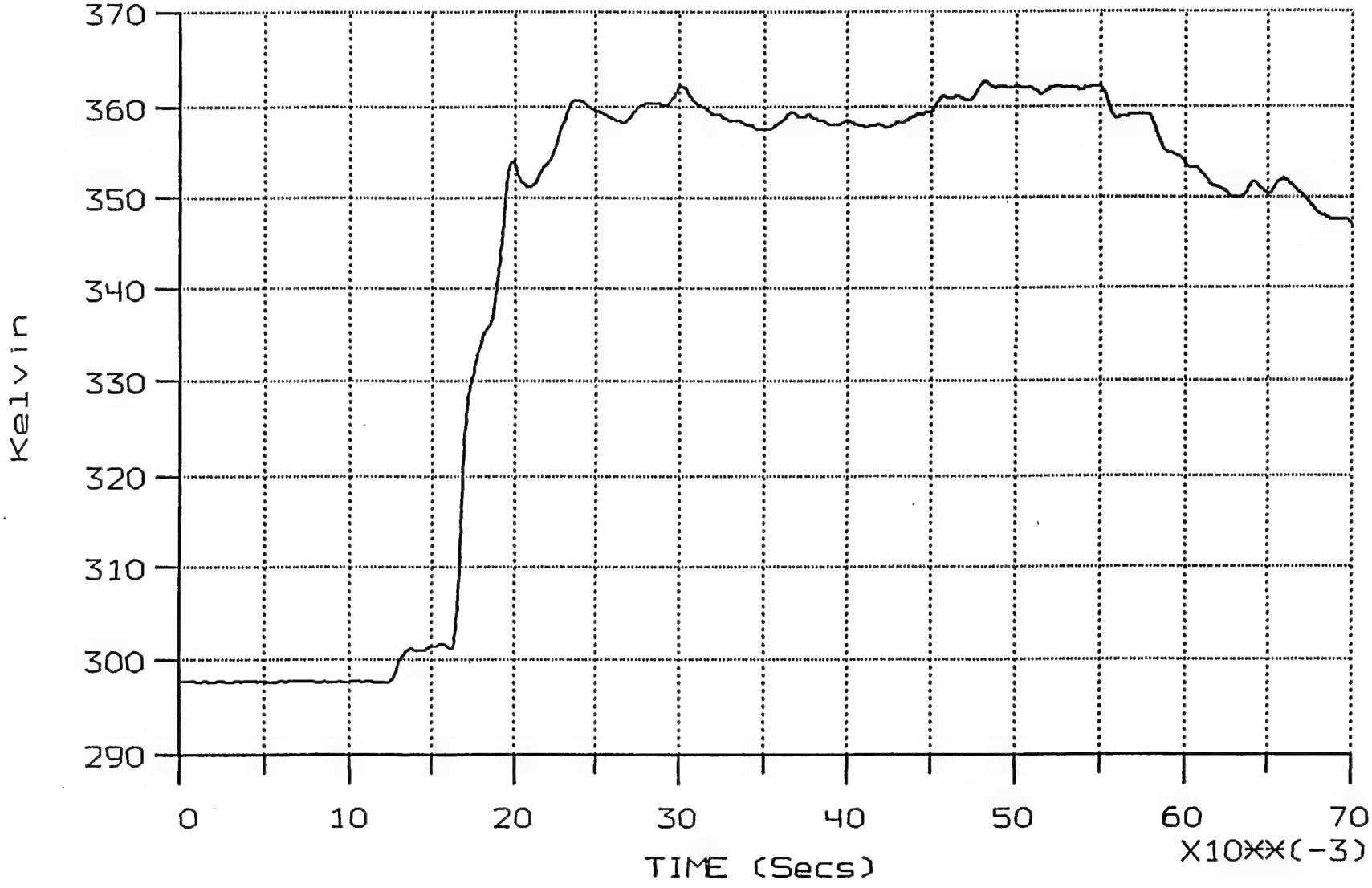


Figure 3.24 -- Surface temperature history for Pyrex-type gauge on SSME turbine

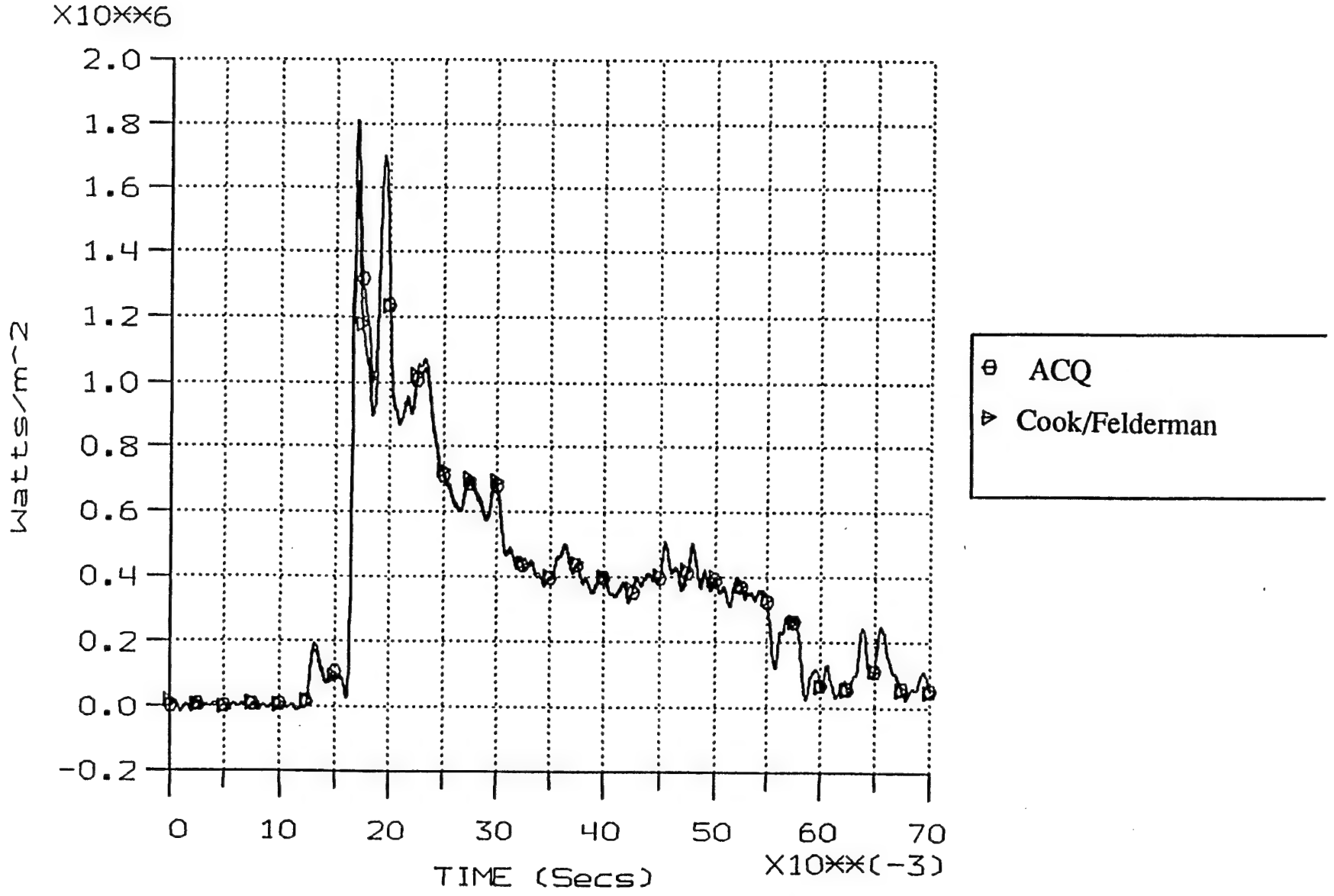


Figure 3.25 -- ACQ and Cook/Felderman calculated heat-flux for Pyrex-type gauge on SSME turbine

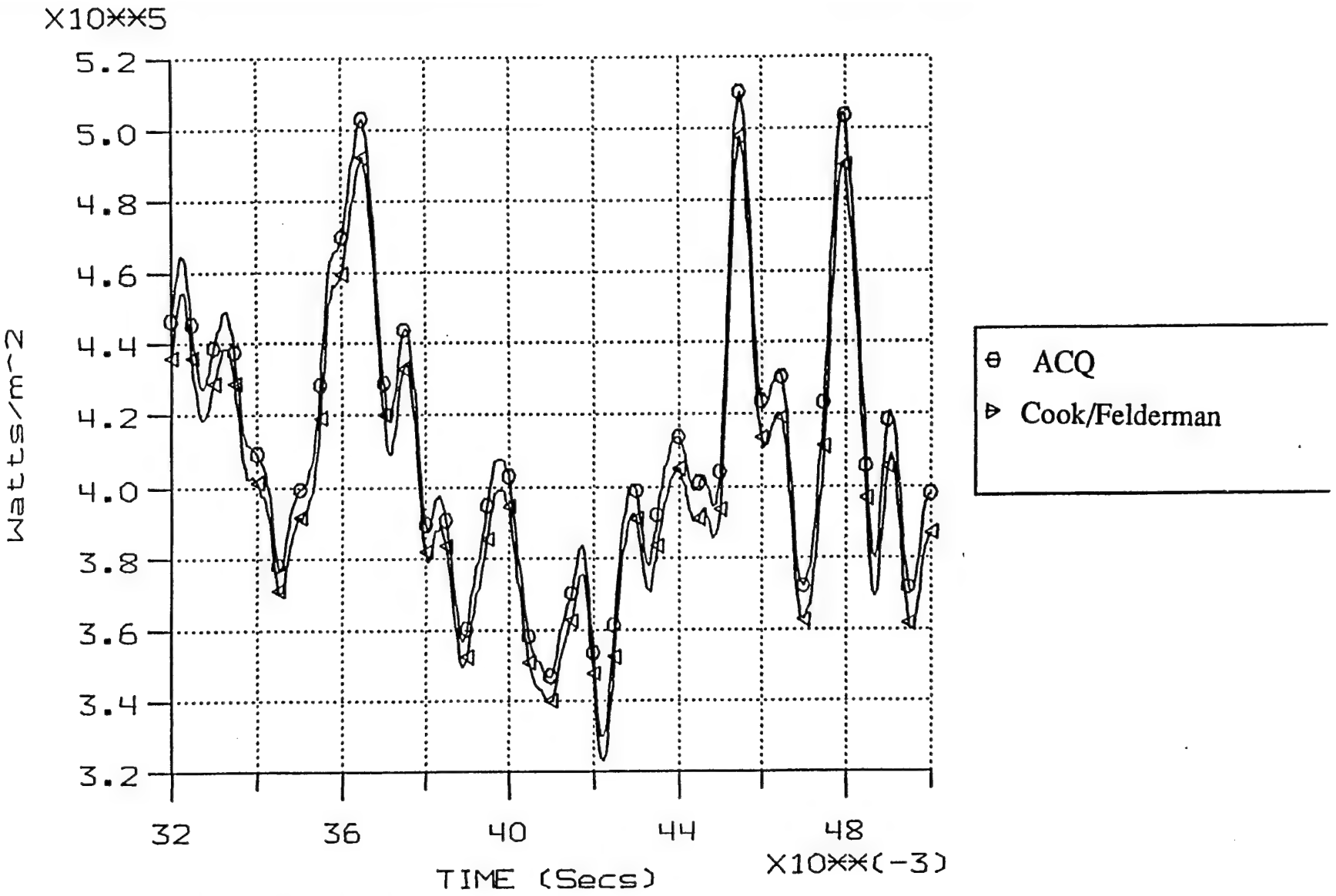


Figure 3.26 -- ACQ and Cook/Felderman calculated heat-flux for Pyrex-type gauge on SSME turbine

SECTION 4 -- CODE APPLICATION

This section provides the user with the information necessary to properly apply ACQ. ACQ is compatible with the Data Acquisition System (DAS). It operates with DAS format input and output files reading and creating the appropriate file headers. The program can be operated in two modes: interactive and automatic. The interactive mode provides the user with a convenient tool for generating heat-flux time histories for one or two gauges. The automatic mode is more like a batch process which not only performs multiple gauge heat-flux calculations, but also places pertinent information into the run log files.

The first part of this section outlines the code's two modes of operation with emphasis on the required input. The second section examines some important considerations in using ACQ. These considerations point out that the data reduction process must be considered in the design of the experiment.

4.1 -- Operation of ACQ

ACQ is fully automated so that multiple gauge heat-flux calculations can be invoked with a single command line. In this automatic mode, the code operation is controlled through a sensor details file. As indicated earlier, the automatic mode can be passed by to give the user greater versatility in an interactive mode. However, the sensor details file can still be utilized within the interactive mode.

4.1.1 -- Sensor Details File

The sensor details file is a user created listing of similar gauges that are all mounted on a given test article. For instance, a sensor details file would typically contain a listing of all the double-sided gauges on a particular turbine with a line in the file for each test run recorded. Each line contains information on the gauge and the run data. The user is free to break up the gauges and runs into as many sensor files as is convenient.

A line in the sensor details file contains the gauge/run label in single quotes, the gauge span and wetted distance coordinates, the $\rho C_p k$ [W^2s/m^4k] and the k/d [W/m^2k] of the gauge. The gauge/run label is the six character DAS designation for the particular test run and sensor. This label is used as a prefix for the input temperature file names; i.e., 'LABEL'T.1 for top temperature file and 'LABEL'B.1 for bottom temperature file. Likewise, the heat-flux output file generated by ACQ is 'LABEL'Q.2.

4.1.2 -- Interactive Mode

Operation of the ACQ interactive mode is outlined below with an explanation the various prompts.

- Type 'acq' to enter interactive mode.
- 'PROCESS A SERIES OF DATA FILES? (Y/N)'.

  'Y': Enter the sensor details file name. Program then allows the user to delete any items in the file not wanted for processing.

  'N': Enter gauge label, $\rho C_p k$, and k/d. Program then allows the user to edit any of the three inputs before beginning.

- 'FINITE LENGTH (0) OR SEMI-INFINITE (1) LINE (0/1)?'

  '0' for double-sided gauge

  '1' for single-sided gauge

- If only one gauge is being processed, the program provides the versatility to use filenames other than the displayed defaults.

  'ENTER FILENAME FOR FIRST UPPER TEMP. [DEF 'LABEL'T.1]'

  'ENTER FILENAME FOR FIRST LOWER TEMP. [DEF 'LABEL'B.1]'

  'ENTER FILENAME FOR FIRST HEAT FLUX [DEF 'LABEL'Q.2]'

- 'ENTER MAXIMUM FREQUENCY (HZ) FOR ANALOGUE [DEF=$f_S/2$]' The default (1/2 sample frequency) is the largest possible frequency allowable before frequency folding occurs. A frequency less than the default may be entered.

- 'ENTER REQUIRED NUMBER OF STAGES [DEF=9]' Experience indicates that 9 analogue nodes is optimum for the program. However, for unusual circumstances this may be changed.

- 'CAPACITOR GEOMETRIC VALUE IS ' γ'. IS THIS ACCEPTABLE (Y/N)? [DEF=Y]' ACQ allows the user to review the calculated geometric factor. If for some reason the value is deemed unacceptable, the user can change the number of stages until the geometric factor is acceptable.

- If a finite length line is used, the program prompts 'ENTER START TIME (MS)'. The start time for a semi-infinite line is automatically zero.

- 'ENTER END TIME (MS)'. This can be less than or equal to the end of the data file.

- 'SCREEN OUTPUT EVERY N STEPS, ENTER N'. ACQ allows the user to view the calculated heat-flux values on the screen.

- ACQ then creates a DAS output file of the specified name with the appropriate header information and heat-flux values.

4.1.3 -- Automatic Mode

The automatic mode allows ACQ to be operated from the Data Reduction Process (DRP) environment through an 'X11' interface. ACQ is invoked with the following command line: 'acq FILENAME [STARTTIME[ENDTIME[I_SINF]]]'.

- 'FILENAME' refers to the sensor details file that contains the gauge/run labels to be reduced.

- 'STARTTIME' is in ms. For a semi-infinite analogue, this is set to zero, and if omitted the default is zero.

- 'ENDTIME' is in ms. If omitted, the default is the end of the data file.

- 'I_SINF' is an integer indicator to select the semi-infinite analogue. If omitted or zero, the finite analogue is used; otherwise, the semi-infinite analogue is used.

When ACQ is accessed from the DRP, the user is informed from the following list of error messages of any error which causes the code to abort:

1.) sensor details file does not exist

2.) error reading sensor details file

3.) required 'INDEXER.1' file does not exist

4.) no data in the specified time interval

5.) upper and lower temperature files recorded differently

6.) start time error

7.) end time error

8.) convergence error in eigenvalue problem

4.2 -- Data Reduction Considerations

This section reemphasizes some of the subtle, yet important, considerations for applying ACQ. For instance, for a finite analogue the upper and lower surface temperatures must correspond to the same time; i.e., simultaneously sampled. Also the finite analogue allows the user to reduce the data over a smaller subinterval of the total test time by choosing a start time other than zero.

In applying ACQ with a semi-infinite analogue, there are two important considerations to review. The first is the semi-infinite assumption which must be valid for the data being reduced. This assumption is met if the test time is less than the reciprocal of the minimum frequency calculated according to Equation 2.30; i.e.,

$$t_{end} - t_{start} < \left\{ \left(\frac{k}{d} \right)^2 \frac{1}{\rho C_p k} \right\}^{-1} \tag{4.1}$$

The second consideration with a semi-infinite analogue is the initial condition. As stated in Section 2.2.3.2.2, the initial condition must be a known uniform substrate temperature. This is accomplished by ensuring that the first data point ($t_{start} = 0$) is prior to any test section flow. The next very important consideration, discussed in more detail below, is the stability of the integration which applies to both finite and semi-infinite analogues.

4.3.1 -- Stability

The fourth order Runge-Kutta integration method used by ACQ is conditionally stable. Hoffman [1990] notes that the following condition must be met for this method to be stable.

$$\alpha(\Delta t) \leq 2.785 \tag{4.2}$$

where α is taken from the homogeneous Ordinary Differential Equation (ODE) of the form

$$\frac{dy}{dt} = -\alpha y(t)$$

The ODE integrated within ACQ is given by Equation 2.13. Its corresponding homogeneous ODE is

$$\frac{dy_i}{dt} = \frac{1}{R_1 C_1}(\lambda_i y_i)$$

thus,

$$\alpha = \frac{-\lambda_i}{R_1 C_1}$$

The stability criterion for ACQ is therefore given by

$$f_s \geq \frac{-\lambda}{(2.785) R_1 C_1} \tag{4.3}$$

where λ is the largest negative eigenvalue for the analogue's system of equations.

Examination of Equation 4.3 indicates that stability is a function of the following input parameters: $\rho C_p k$, k/d, no. of nodes, and maximum frequency. Unfortunately, Equation 4.3 is not simple to evaluate because it requires the eigenvalues of the analogue's system of equations. To illustrate some typical minimum sample frequencies, Equation 4.3 is evaluated for the two gauges used in the Section 3 verification tests.

The double-sided Kapton gauge minimum sample frequency for stability is evaluated using the steady harmonic heat-flux test number 1. The parameters for this test are given in Table 3.1, of Section 3, with f_{max} = 50 kHz and n = 9. The nine eigenvalues range from -1.6418 to -0.014017. Therefore, the minimum sample frequency for this case is 89 kHz.

The minimum sample frequency for the single-sided Pyrex gauge is evaluated using the convective heat transfer test case. The parameters for this test case are given in Table 3.3 with f_{max} = 10 kHz and n = 9. The eigenvalues range from -1.3548 to -9.8002×10^{-6}. Evaluating Equation 4.3 gives a minimum frequency of 13 kHz.

In both cases presented here, the minimum sample frequency for stability is less than the minimum sample frequency required to prevent frequency folding. However, caution should be taken in extending these results to dissimilar gauges. In fact, it might be necessary to generate a test case for the particular gauge and sample frequency to ensure that the code will be stable.

5. LIST OF REFERENCES

Burd, H. J. and Doe, N. G., 1980, "The Computer Analysis, and Design of, an Electrical Analogue of the Heat Conduction Process for use in a Transient Cascade to Obtain Heat Transfer Data from Temperature Measurements", Third year project under M. L. G. Oldfield at Oxford University.

Cook, W. J. and Felderman, E. J., 1966, "Reduction of Data from Thin-Film Heat-Transfer Gages: A Concise Numerical Technique", *AIAA Journal,* Vol. 4, No. 3, pp. 561 & 562.

Epstein, A. H., Guenette, G. R., Norton, R. J. G., and Yuzhang, C., 1986, "High-frequency response heat-flux gauge", *Rev. Sci. Instrum.* 57, pp. 639-649.

Hoffman, J. D., 1990, *Numerical Methods for Engineers and Scientists,* John Wiley & Sons, New York, New York.

Incropera, F. P. and DeWitt, D. P., 1985, *Introduction to Heat Transfer,* John Wiley & Sons, New York, New York.

Norton, R. J. G., 1985, "What ACQ Does", Unpublished notes for USAF and MIT distribution.

Skinner, G. T., 1960, "Analog Network to Convert Surface Temperature to Heat Flux", Cornell Aeronautical Laboratory, Report No. CAL-100, Buffalo, New York.

APPENDIX -- ACQ CODE

```
      program acq

*         Simulates a logarithmic high-bandwidth resistance-capacitance
*     (R-C) analogue to obtain high frequency heat transfer rates.

*     Theory from:

*     Oxford University Engineering Laboratory Report 1382/81
*     Authors: M.L.G.Oldfield, H.J.Burd and N.G.Doe
*     "Design of Wide-Bandwidth Analogue Circuits for Heat Transfer
*      Instrumentation in Transient Tunnels"

*     Version 1.10 R.J.G.Norton.....Oct-30-1984

*         Modified so that user specifies number of stages and the
*     capacitor geometric factor is calculated.
*         Further modified (Fall 1993) to correct initial boundary
*     conditions and enable I/O of DAS format files.

*         The program has been modified to operate in two modes:
*     interactive and automatic.  The interactive mode provides the
*     user with a convenient tool for generating heat flux time
*     histories for one or two sensors.  It is run simply by entering
*     the command "acq" and supplying requested information.
*         The automatic mode is more like a batch process and in
*     addition to the heat flux attends to placing pertinent information
*     in turbine run log files.  The form of the command to execute the
*     program in automatic mode is:
*         "acq filename [ mintim [ maxtim [ i_sinf ] ] ]".
*     Where, filename refers to the required sensor details file; mintim
*     and maxtim are respective min & max times in msecs; and, i_sinf is
*     an integer indicator to select the semi-infinite length analogue.
*     if i_sinf is missing or zero, the finite length analogue is used.
*     Otherwise, the semi-infinite assumption is used and mintim = zero.
*     The sensor details file consists of one or more lines containing,
*     the gauge label in single quotes; x,y coordinates; and rho*c*k
*     (W^2-s/m^4-K)  & k/d (W/m^2-K) of the substrate material.  The
*     properties of the first gauge in the sensor details files are used
*     to determine global treatment of the rest.  If mintim & maxtim are
*     not given, the entire recorded time interval of the first gauge is
*     used.
*         Since the data reduction process (drp) accesses acq, there
*     must be some way of informing the user of error conditions.  A
*     list of the conditions which cause the code to abort follows:
*         1. sensor details file does not exist;
*         2. error reading sensor details file;
*         3. required "INDEXER.1" file does not exist;
*         4. no data in the specified time interval;
*         5. upper and lower temp files recorded differently;
*         6. mintim error;
*         7. maxtim error;
*         8. minfrq error;
*         9. convergence error in valvar.

*************
```

```
*      Work Arrays (Size):

*      CAP(MAXST)        - Capacitance values for each part of the network
*      RES(MAXST+1)      - Resistance values for each part of the network
*      D(MAXST)          - B matrix storage - diagonal elements
*      E(MAXST)          - B matrix storage - off-diagonal elements
*      A(MAXST,MAXST)    - Eigen vector storage
*      Y(MAXST)          - Solution at each node (transformed voltage)
*      XKOD(MAXPTS)      - K/D for each gauge
*      RCK(MAXPTS)       - RHO*C*K for each gauge

*************

#include <acqchg.inc>
#include <acqfil.inc>
#include <acqrpa.inc>
#include <acqsiz.inc>
#include <dstruc.inc>
      record /rundes/ trdr, brdr, qrdr
      record /chdes / tcdr, bcdr, qcdr

      parameter  (maxst = 40)
      character   fname*9, fpos*70, foutt*70, foutb*70, foutq*70
      character   datstr*9, fnp0*6/'(fn.0)'/, iw*4/'(iw)'/, timstr*8,
     +            usrnam*10,
     +            chirun*6, ch_out(4)*10, finput*70, logfil*13
      integer     getlc, iargc, nc_out(4)
      logical     exists, screen
      pointer    (ipvalt,vt), (ipvalb,vtb), (ipvalo,vto), (iptime,rt)
      real*4      vt(*), vtb(*), vto(*), rt(*), cvalue(4)
      real*8      a(40,40), cap(40), d(40), e(40), res(41), t(40),
     1            y(40), pi, r8time(128), sumres, twopi,
     2            amp, an, beta, bn, c1, cn, ctot, deltat, dgam, dn, dt,
     3            ffunc, fmax, fmin, gam, gamn, r1, rr1c1, rtot, vol1,
     4            time,
     5            u1, v0, v0b, v1, v1b, v5, v5b, vfor5b, vforc1, vforc5,
     6            vforce, vfor1b, vforeb

*      Logical Unit Numbers:
*      1 - DAS upper temperature file
*      2 - DAS lower temperature file
*      3 - Q vs time output file
*      4 - Run log file
*     19 - Internal "scratch" file

      pi     = dacos(-1d0)
      twopi  = 2d0 * pi
      call getenv('USER',usrnam)
      open(19,status='scratch')
      ipvalt = 0
      ipvalb = 0
      ipvalo = 0
      iptime = 0
      n_args = iargc()
      screen = .true.
      if(n_args.gt.0) then
```

```
        screen = .false.
        call getarg(1,finput)
      endif

*     Get descriptor names, RHO*C*K and K/D
      call readna(0,0,1,1,npos,screen,finput)

*     Determine whether finite or semi-infinite length analogue

      if(screen) then
        isemi  = -1
        numerr = 0
        do while (isemi.lt.0 .or. isemi.gt.1)
  100     write(*,'(/a,$)')
     +   'FINITE LENGTH (0) OR SEMI-INFINITE (1) LINE (0/1)? '
          read(*,*,err=100,end=200) isemi
        enddo
      endif

*     Set default data file names

      fname  = name(1)
      ncstem = lnblnk(fname)
      foutt  = fname(:ncstem)//'T.1'
      foutb  = fname(:ncstem)//'B.1'
      foutq  = fname(:ncstem)//'Q.2'
      ncf    = ncstem + 3

*     Prompt for filenames if only one gauge is being treated

      if(npos.eq.1) then
        numerr = 0
        if(screen) then
          write(*,101) 'UPPER TEMP.',foutt(:ncf)
  101     format(/'ENTER FILENAME FOR FIRST ',A,' [DEF ',A,']')
          read(*,'(a)',end=200) fpos
          if(fpos.ne.' ') foutt = fpos
          if(isemi.eq.0) then
            write(*,101) 'LOWER TEMP.',foutb(:ncf)
            read(*,'(a)',end=200) fpos
            if(fpos.ne.' ') foutb = fpos
          endif
          write(*,101) 'HEAT FLUX',foutq(:ncf)
          read(*,'(a)',end=200) fpos
          if(fpos.ne.' ') foutq  = fpos
        endif
      endif

********************  Set up network requirements **********************

*     First: set number of stages and capacitance/resistance variation
*     along the line.

      open(1,file=foutt,status='old',form='unformatted',
     +     recl=1024,access='direct')
      read(1,rec=1) trdr
```

```
      read(1,rec=2) tcdr
      close(1,status='keep')

      if(screen) then
        numerr = 0
*       Input maximum analogue frequency
  110   fmax   = tcdr.lsr / 2
        write(*,'(a,1pe11.4,a,$)')
     + 'ENTER MAXIMUM FREQUENCY (HZ) FOR ANALOGUE [DEF=',fmax,'] '
        read(*,'(a)',end=200) fpos
        if(fpos.ne.' ' .and. fpos.ne.'/') then
          rewind 19
          write(19,'(a)') fpos(:lnblnk(fpos))
          rewind 19
          read(19,*,err=110) fmax
          if(fmax.le.0.0) go to 110
        endif
      else
        inprun = trdr.run
        call num2ch(inprun,chirun,nchrun)
        logfil = 'run'//chirun(:nchrun)//'.log'
        inquire(file=logfil,exist=exists)
        if(exists) then
          open(4,file=logfil,access='append')
        else
          open(4,file=logfil,status='new')
        endif
        call ufdate(datstr,timstr)
        fmax   = tcdr.lsr / 2
        deltat = 1d0 / tcdr.lsr
        st     = 0.0
        ft     = (tcdr.nsampl - 1) * deltat
        isemi  = 0
        if(n_args.gt.1) then
*         Second argument is start time in msecs
          call getarg(2,fpos)
          nfp    = lnblnk(fpos)
          rewind 19
          write(19,'(a)') fpos(:nfp)
          rewind 19
          fnp0(3:3) = char(nfp+48)
          numerr = 6
          read(19,fnp0,err=200) st
          st     = st / 1d3
          if(n_args.gt.2) then
*           Third argument is final time in msecs
            call getarg(3,fpos)
            nfp    = lnblnk(fpos)
            rewind 19
            write(19,'(a)') fpos(:nfp)
            rewind 19
            fnp0(3:3) = char(nfp+48)
            numerr = 7
            read(19,fnp0,err=200) ft
            ft     = ft / 1d3
            if(n_args.gt.3) then
```

```
*             Fourth argument is minimum frequency in HZ
              call getarg(4,fpos)
              nfp    = lnblnk(fpos)
              rewind 19
              write(19,'(a)') fpos(:nfp)
              rewind 19
              iw(3:3) = char(nfp+48)
              numerr = 8
              read(19,iw,err=200) isemi
              if(isemi.ne.0) then
                isemi  = 1
                fmin   = xkod(1)**2 / (rck(1) * twopi)
                st     = 0d0
              endif
            endif
          endif
        endif
        write(4,'(a8,2x,a9,2x,a10,2x,a)') timstr,datstr,usrnam,
     +                                    'Program acq specifics :'
        ncfin  = lnblnk(finput)
        write(4,'(a)') '* Sensor details taken from '//finput(:ncfin)
        write(4,'(a,1pe11.4,a)') '* with RHO*C*K of the substrate =',
     +                           rck(1),' W^2-s/m^4-K^2.'
        if(isemi.eq.0) then
          write(4,'(a,1pe11.4,a)')
     +   '* The finite length analogue with K/D =',xkod(1),
     +   ' W/m^2-K was used.'
        else
          write(4,'(a)')
     +   '* The semi-infinite length analogue with K/D =',xkod(1),
     +   ' W/m^2-K was used.'
        endif
        cvalue(1) = st * 1000
        cvalue(2) = ft * 1000
        cvalue(3) = fmin
        cvalue(4) = fmax
        do i=1,4
          write(fpos,'(f10.3)') cvalue(i)
          if(cvalue(i).eq.0.0) then
            ch_out(i) = '0'
            nc_out(i) = 1
          else
            i1      = ldblnk(fpos)
            i2      = lnblnk(fpos)
            if(cvalue(i).eq.aint(cvalue(i))) then
              i2      = i2 - 4
            else
              do while (fpos(i2:i2).eq.'0')
                i2      = i2 - 1
              enddo
            endif
            ch_out(i) = fpos(i1:i2)
            nc_out(i) = i2 - i1 + 1
          endif
        enddo
        write(4,'(a)') '* Time interval = ['//ch_out(1)(:nc_out(1))//
```

```
     +                  ',''//ch_out(2)(:nc_out(2))//
     +                  '] msecs; Min & Max Frequency = '//
     +                  ch_out(3)(:nc_out(3))//' & '//
     +                  ch_out(4)(:nc_out(4))//' Hz.'
      endif

*     Check for finite or semi-infinite line

      if(isemi.eq.0) then
*       Finite length
        if(screen) write(*,'(/a/)')
     + 'USING FIRST GAUGE PROPERTIES TO DEFINE NORMALIZED ANALOGUE.'
        rhocd  = rck(1) / xkod(1)
*       Set gamma determining forcing function
        ffunc  = twopi * fmax * rhocd / xkod(1)
        ffunc  = dsqrt(ffunc)
      else
*       Semi-infinite length
        if(screen) then
          numerr = 0
  120     fmin   = xkod(1)**2 / (rck(1) * twopi)
          write(*,'(a,1pe11.4,a,$)')
     +   'ENTER MINIMUM FREQUENCY (HZ) FOR ANALOGUE [DEF=',fmin,'] '
          read(*,'(a)',end=200) fpos
          if(fpos.ne.' ' .and. fpos.ne.'/') then
            rewind 19
            write(19,'(a)') fpos(:lnblnk(fpos))
            rewind 19
            read(19,*,err=120) fmin
          endif
*         Check for legal frequency limits
          if(fmin.ge.fmax) then
            write(*,'(a)') 'FMIN MUST BE LESS THAN FMAX.  TRY AGAIN.'
            go to 120
          endif
          if(fmin.le.0d0) then
            write(*,'(a)') 'FMIN MUST BE GREATER THAN ZERO.  TRY AGAIN.'
            go to 120
          endif
        endif
*       Set gamma determining forcing function
        ffunc  = dsqrt(fmax/fmin)
      endif

*     Now for both get required number of stages

  125 if(screen) then
        numerr = 0
  130   nst    = 9
        write(*,'(a,i2,a,$)')
     + 'ENTER REQUIRED NUMBER OF STAGES [DEF=',nst,'] '
        read(*,'(a)',end=200) fpos
        if(fpos.ne.' ' .and. fpos.ne.'/') then
          rewind 19
          write(19,'(a)') fpos(:lnblnk(fpos))
          rewind 19
```

```
          read(19,*,err=130) nst
        endif
        if(nst.gt.maxst) then
          write(*,'(/2(a,i5),a)')
     +   'NUMBER OF STAGES IS ',nst,'.  MAXIMUM IS ',maxst,'.'
          go to 130
        endif
      else
        nst    = 9
      endif

*     Solve iteratively for gamma

      gam    = 1.4d0
      dgam   = 1d0
      do while (dgam.gt.1d-10)
        gamn   = 1d0 - ffunc * (1d0 - gam) / (1d0 + dsqrt(gam))
        gamn   = gamn**(1d0/nst)
        dgam   = dabs(gam-gamn)
        gam    = gamn
      enddo

      beta   = 1d0 / (1d0 + dsqrt(gam))

      if(screen) then
        numerr = 0
*       Inform user of gamma value and ask if okay
  140   write(*,'(a,1pe13.6,a)') 'CAPACITOR GEOMETRIC VALUE IS ',gam,
     +                           ' IS THIS ACCEPTABLE (Y/N)? [DEF=Y]'
        i      = get1c()
        if(i.eq.78 .or. i.eq.110) go to 125     ! N or n
        if(i.ne.13 .and. i.ne.89 .and. i.ne.121) go to 140
      endif

*****************************************************************
*     The number of stages and their distribution is now known.
*     Set network -- non-dimensionalize by the first resistor and
*     capacitor values.  These real values will be introduced when
*     the governing equations are solved.
*     Note:  This structure assumes that the same "spacing" of
*            elements will be used for all cases run, and thickness of
*            the kapton/property variations will be introduced when
*            each of the cases to be run is considered (i.e. through
*            R1 and C1 since total resistance = D/K, and
*                           total capacitance = RHO*C*D).
*****************************************************************

*     Now set the first capacitor/resistor values to 1.  The network
*     equations will become non-dimensional to R1*C1, and R1*C1 will be
*     introduced when the time steps are done.
*
      cap(1) = 1D0
      res(1) = 1D0

*     Fill line resistance/capacitance values with R/R1 and C/C1
```

```
*       Note, in real terms:
*       C(I) = GAM*C(I-1)
*       R(1) = (R/C)*C1/(1+SQRT(GAM))
*       R(I) = (R/C)*CI/SQRT(GAM)
*       R(NST+1) = (R/C)*CNST*SQRT(GAM)/(1+SQRT(GAM)) - finite length
*       R(NST+1) = infinite for semi-infinite

        do i=2,nst
          cap(i) = gam * cap(i-1)
          res(i) = dsqrt(gam) * (1d0 + dsqrt(gam)) * cap(i-1)
        enddo
        res(nst+1) = dsqrt(gam)*cap(nst)
        if(isemi.eq.1) res(nst+1) = 1d38

*       Get total resistance and capacitance (normalized by R1,C1)

        rtot    = 0D0
        ctot    = 0D0
        do i=1,nst
          rtot    = rtot + res(i)
          ctot    = ctot + cap(i)
        enddo
        if(isemi.eq.0) rtot    = rtot + res(nst+1)

*****************************************************
*       Now fill the B-matrix (i.e. D and E matrices)
*****************************************************

*       Note: Bi,i = - 1/Ci * (1/Ri + 1/Ri+1)
*             Bi,i+1 = Bi+1,i = 1/(Ri+1*SQRT(Ci*Ci+1))

        do i=1,nst
          d(i) = -(1d0 / res(i) + 1d0 / res(i+1)) / cap(i)
          if(i.lt.nst) e(i+1) = 1d0 / (res(i+1)*dsqrt(cap(i)*cap(i+1)))
        enddo

*       Set input transformation matrix A to identity matrix.
*       (On return from VALVAR it contains the eigenvectors.)

        do j=1,maxst
          do i=1,maxst
            a(i,j) = 0d0
          enddo
          a(j,j) = 1d0
        enddo

*       Now get eigenvector matrix.  Eigenvalues are stored in matrix D,
*       hence, D is destroyed.

        call valvar(d,e,a,ic,nst,maxst)

        if(ic.ne.0) then
          numerr = 9
          if(screen)
     +    write(*,'(a)') 'Convergence error in VALVAR -- stopping.'
          go to 200
```

```
      endif

*****************************************************************
*     This completes the section on setting up the non-dimensional
*     analogue network.  Now we get into dimensional numbers and
*     solve the equations.
*****************************************************************

*****************************************************************
*     Get physical properties so that analogue really simulates the
*     heat transfer environment.
*****************************************************************

*     Get RHO*C*K if semi-infinite or K/D and RHO*C*D if finite length.
*     (i.e. Input voltage = temperature => input current = heat
*           transfer rate.)

      if(screen) then
        numerr = 0
        if(isemi.eq.0) then
*         First get start time
  150     write(*,'(a,$)') 'ENTER START TIME (MS) => '
          read(*,*,err=150,end=200) st
          st      = st / 1d3
        else
          st      = 0d0
          write(*,'(a)') 'START TIME = 0 MSECS.'
        endif
*       Get end time
  160   write(*,'(a,$)') 'ENTER END TIME (MS) => '
        read(*,*,err=160,end=200) ft
        ft      = ft / 1d3
*       Get screen print interval
  170   write(*,'(a,$)') 'SCREEN OUTPUT EVERY N STEPS, ENTER N => '
        read(*,*,err=170,end=200) ipr
        if(ipr.le.0) ipr    = 1
        write(*,'(/)')
      endif

*     Read through the files and get data

      do nnn=1,npos

*       Set driving voltages to zero
        v0      = 0d0
        v1      = 0d0
        v0b     = 0d0
        v1b     = 0d0
        v5b     = 0d0
        vforeb  = 0d0
        vfor1b  = 0d0
        vfor5b  = 0d0

        if(npos.gt.1) then
*         Set temperature data file names
          fname  = name(nnn)
```

```
          ncstem = lnblnk(fname)
          foutt  = fname(:ncstem)//'T.1'
          foutb  = fname(:ncstem)//'B.1'
          foutq  = fname(:ncstem)//'Q.2'
        endif

        if(screen) then
*         Tell user the files we are reading and writing
          write(*,'(a)') 'READING FILE:- '//foutt(1:length(foutt))
          if(isemi.eq.0) write(*,'(a)') 'READING FILE:- '//
     +                                   foutb(1:length(foutb))
          write(*,'(a)') 'WRITING FILE:- '//foutq(1:length(foutq))
        endif

*       Open upper surface temperature file for read
        open(1,file=foutt,status='old',form='unformatted',
     +       recl=1024,access='direct')

        read(1,rec=1) trdr
        read(1,rec=2) tcdr

*       Compute required properties
        if(isemi.eq.0) then
*         Finite-length
          rhocd=rck(nnn)/xkod(nnn)
          r1 = 1d0/(xkod(nnn)*rtot)
          c1 = rhocd/ctot
        else
*         Semi-infinite length
*         Compute first stage capacitance and resistance
          c1 = 1d0/dsqrt(twopi*fmax*beta*beta/rck(nnn))
          r1 = 1d0/(twopi*fmax*beta*c1)
        endif
        rr1c1=1d0/(r1*c1)

        ichnum = ichar(tcdr.lss)
        if(ichnum.gt.96) tcdr.lss = char(ichnum-32)
        ntimes = tcdr.nsampl
        if(ipvalt.gt.0) call free(ipvalt)
        ipvalt = malloc(ntimes*4)
        ntrecs = 2 + (ntimes + 255) / 256
        j2     = 0
        do irec=3,ntrecs
          j1     = j2 + 1
          j2     = min0(j2+256,ntimes)
          read(1,rec=irec) (vt(j),j=j1,j2)
        enddo
        close(1,status='keep')
        if(iptime.gt.0) call free(iptime)
        iptime = malloc(ntimes*4)
        if(tcdr.lss.eq.'I') then
          inquire(file='INDEXER.1',exist=exists)
          if(exists) then
            open(1,file='INDEXER.1',status='old',access='direct',
     +           form='unformatted',recl=1024)
            nirecs = 2 + (ntimes + 127) / 128
```

```
            j2      = 0
            j0      = 0
            jn      = 0
            do irec=3,nirecs
              j1      = j2 + 1
              j2      = min0(j2+128,ntimes)
              ninput = j2 - j1 + 1
              read(1,rec=irec) (r8time(j),j=1,ninput)
              k       = 0
              do j=j1,j2
                k       = k + 1
                rt(j)   = r8time(k)
                if(rt(j).le.st) j0      = j
                if(rt(j).le.ft) jn      = j
              enddo
            enddo
            close(1,status='keep')
          else
            numerr = 3
            if(screen) write(*,'(a)')
     +                 'FILE "INDEXER.1" DOES NOT EXIST -- STOPPING.'
            go to 200
          endif
        elseif(tcdr.lss.eq.'C') then
          deltat = 1d0 / tcdr.lsr
          j0      = 1.5 + st / deltat
          jn      = 1.5 + ft / deltat
          j0      = max0(1,j0)
          jn      = min0(ntimes,jn)
          do j=1,ntimes
            rt(j) = (j - 1) * deltat
          enddo
        endif

*       Check for meaningful time interval indices

        numerr = 4
        if(j0.lt.1 .or. j0.gt.ntimes .or. j0.ge.jn) go to 200
        if(jn.lt.1 .or. jn.gt.ntimes) go to 200
        nud     = jn - j0 + 1
        nub     = jn

*       Open heat transfer rate file for write

        open(3,file=foutq,status='unknown',form='unformatted',
     +         recl=1024,access='direct')
        qrdr    = trdr
        qcdr    = tcdr
        qcdr.unityp = 5
        qcdr.unilab = 'Watts/m^2'
        qcdr.ctype = 2
        nclabl = lnblnk(tcdr.label)
        qcdr.label(nclabl:nclabl) = 'Q'
        nstory = min0(lnblnk(qcdr.story),66)
        qcdr.story = qcdr.story(:nstory)//' QDOT FROM ACQ'
        write(3,rec=1) qrdr
```

```
      write(3,rec=2) qcdr
      if(ipvalo.gt.0) call free(ipvalo)
      ipvalo = malloc(ntimes*4)

      if(isemi.eq.0) then
*       Open lower temperature file for read
        open(2,file=foutb,status='old',form='unformatted',
     +           recl=1024,access='direct')
        read(2,rec=1) brdr
        read(2,rec=2) bcdr
        ichnum = ichar(bcdr.lss)
        if(ichnum.gt.96) bcdr.lss = char(ichnum-32)
        if(bcdr.lss.ne.tcdr.lss .or. bcdr.nsamp1.ne.tcdr.nsamp1) then
          numerr = 5
          if(screen) write(*,'(a)')
     +     'TOP/BOTTOM TEMPERATURE FILE INCOMPATIBILITY -- STOPPING.'
          go to 200
        endif
        if(ipvalb.gt.0) call free(ipvalb)
        ipvalb = malloc(ntimes*4)
        j2     = 0
        do irec=3,ntrecs
          j1     = j2 + 1
          j2     = min0(j2+256,ntimes)
          read(2,rec=irec) (vtb(j),j=j1,j2)
        enddo
        close(2,status='keep')
      endif

*     Set boundary conditions based on length analogue chosen

      if(isemi.eq.0) then
*       Finite
        sumres = 0d0
        do m=1,nst
          sumres = sumres + res(m)
          t(m) = vt(j0) + (vtb(j0) - vt(j0)) * r1 * xkod(nnn) * sumres
        enddo
        do m=1,nst
          y(m) = 0d0
          do i=1,nst
            y(m) = y(m) + a(i,m) * dsqrt(cap(i)) * t(i)
          enddo
        enddo
      else
*       Semi-infinite
        do m=1,nst
          y(m) = 0d0
          do i=1,nst
            y(m) = y(m) + a(i,m) * dsqrt(cap(i))
          enddo
          y(m) = vt(j0) * y(m)
        enddo
        v0b    = vt(j0)
        v1b    = v0b
        v5b    = .5d0 * (v0b + v1b)
```

```
            vforeb = v0b/(res(nst+1)*dsqrt(cap(nst)))
            vforlb = v1b/(res(nst+1)*dsqrt(cap(nst)))
            vfor5b = v5b/(res(nst+1)*dsqrt(cap(nst)))
          endif

*         Set up loop counter (IOUT = output file counter)
          iout   = 0

*         Loop for picking up time and values
*           (use upper surface temperature file as timing file)

          if(nud.eq.0) then
*           Inform user there are no data
            if(screen) write(*,'(a)')
     +     'NO TEMPERATURE VALUES IN REQUESTED TIME INTERVAL.'
          else
            u1     = 0d0
            do i=1,nst
              u1     = u1 + a(1,i) * y(i)
            enddo
            vto(j0) = (vt(j0) - u1 / dsqrt(cap(1))) / r1
            do iq=j0+1,jn
              time   = rt(iq)
              v0     = vt(iq-1)
              v1     = vt(iq)
*             Get time step
              dt     = rt(iq) - rt(iq-1)
*             If finite length line, get lower temp at this and last time
              if(isemi.eq.0) then
                v0b    = vtb(iq-1)
                v1b    = vtb(iq)
              endif

*             Now we have time and value(s) so solve for Qdot
*             ***********************************************
*             Assume that voltage varies linearly between samples
*             ***********************************************
              v5     = .5d0 * (v0 + v1)
              vforce = v0/(res(1)*dsqrt(cap(1)))
              vforc1 = v1/(res(1)*dsqrt(cap(1)))
              vforc5 = v5/(res(1)*dsqrt(cap(1)))
              if(isemi.eq.0) then
                v5b    = .5d0 * (v0b + v1b)
                vforeb = v0b/(res(nst+1)*dsqrt(cap(nst)))
                vforlb = v1b/(res(nst+1)*dsqrt(cap(nst)))
                vfor5b = v5b/(res(nst+1)*dsqrt(cap(nst)))
              endif

*             Now march through all the nodes to get solution at time n+1

              do i=1,nst
*               Use a fourth-order Runge-Kutta operation
*               dy(I)/dt = Eigenvalue(I)*y(I) + Eigenvector(1,I)*VFORCE
*                        + Eigenvector(N,I)*VFOREB
*                        = f(t,y)  (If finite length line)
*               First step:  An = dt*f(t,y)
```

```
*             Second step: Bn = dt*f(t+dt/2,y+An/2)
*             Third step:  Cn = dt*f(t+dt/2,y+Bn/2)
*             Fourth step: Dn = dt*f(t+dt,y+Cn)
*             Last step:   y(t+dt) = y(t) + (An + 2Bn + 2Cn + Dn)/6
              an      = rr1c1 * dt * (d(i) * y(i) + a(1,i) * vforce
     +                + a(nst,i) * vforeb)
              bn      = rr1c1 * dt * (d(i) * (y(i) + .5d0 * an)
     +                + a(1,i) * vforc5 + a(nst,i) * vfor5b)
              cn      = rr1c1 * dt * (d(i) * (y(i) +.5d0 * bn)
     +                + a(1,i) * vforc5 + a(nst,i) * vfor5b)
              dn      = rr1c1 * dt * (d(i) * (y(i)+cn) + a(1,i) * vforc1
     +                + a(nst,i) * vfor1b)
              y(i) = y(i) + (an + 2d0 * (bn + cn) + dn) / 6d0
            enddo

*           Now transform back to get voltage at station 1

            u1      = 0d0
            do i=1,nst
              u1      = u1 + a(1,i) * y(i)
            enddo
            vol1    = u1 / dsqrt(cap(1))
*           And to current (i.e. heat transfer rate)
            amp     = (v1 - vol1) / r1
*           Prepare output file array
            vto(iq) = amp
            if(screen) then
              iout    = iout + 1
              if(mod(iout,ipr).eq.0) write(*,'(2(a,1pe14.6),a)')
     +        'Time =',time,' secs;  Heat-flux=',amp,' W/sq m.'
            endif

*           Loop back for next value
          enddo

          do j=1,j0-1
            vto(j) = 0.0
          enddo
          do j=jn+1,ntimes
            vto(j) = 0.0
          enddo
          j2      = 0
          do irec=3,ntrecs
            j1      = j2 + 1
            j2      = min0(j2+256,ntimes)
            write(3,rec=irec) (vto(j),j=j1,j2)
          enddo
          close(3,status='keep')

          if(.not.screen) then
*           Write to Log file
            if(isemi.eq.0) then
              write(4,'(a)') '* Heat flux '//foutq(:lnblnk(foutq))//
     +                       ' from '//foutt(:lnblnk(foutt))//
     +                       ' and '//foutb(:lnblnk(foutb))
            else
```

```
            write(4,'(a)') '* Heat flux '//foutq(:lnblnk(foutq))//
     +                     ' from '//foutt(:lnblnk(foutt))
          endif
        endif

      endif

*     Loop back for next file

    enddo

*   Fini

200 i      = ieee_flags("clear","exception","all",fpos)

    call exit(numerr)

    end
```